AD-A046 806 MICHIGAN UNIV ANN ARBOR GRADUATE SCHOOL OF BUSINESS--ETC F/G 9/2
DATA TRANSLATOR VERSION IIA, RELEASE 2 USER MANUAL REVISION 1.(U)
MAR 77 E KINTZER, J BODWIN, W COOL DCA100-75-C-0064
UNCLASSIFIED WP-DT-3.4 CCTC-CSM-UM-236-77-REV-1 NL

1 OF 4
AD A046806

AD A046806

DDC

AD NO. ____
DDC FILE COPY

COMPUTER SYSTEM MANUAL
CSM UM 236-77
MARCH 1977

2 B.S.

COMMAND
& CONTROL
TECHNICAL
CENTER

DDC
RECEIVED
NOV 28 1977
B

DATA TRANSLATOR
VERSION IIA, RELEASE 2
USER MANUAL
(REVISION 1)

USERS MANUAL

DEFENSE
COMMUNICATIONS
AGENCY

Approved for public release;
distribution unlimited.

(See 1473)

Unclassified

4 SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(18) CTC

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (19) CSM-UM-236-77-REV-1	2. GOVT ACCESSION NO. (9) Working Paper
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) (6) DATA TRANSLATOR VERSION IIA RELEASE 2 USER MANUAL (Revision 1)	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER (14) WP-DT-3.4
7. AUTHOR(s) (10) Eric Kintzer, James Bodwin, William Cool, Linda Hutchins, Andrew Marine, ~~Kenneth Moore, Don Shemanske, Gregg Wolfe~~	8. CONTRACT OR GRANT NUMBER(s) (15) DCA 100-75-C-0064
9. PERFORMING ORGANIZATION NAME AND ADDRESS (1) ~~Data Translation Project~~ (3) Grad Sch of Bus Admin, 276 ~~Business Administration~~, Univ of Michigan, Ann Arbor, ~~Michigan~~ 48109	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 32017K, ~~27400~~, 27402
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Communications Agency, 11440 Isaac Newton Square, N. Reston, Virginia 22090	12. REPORT DATE (11) Mar[redacted]77
	13. NUMBER OF ~~PAGES~~ (12) 363p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same as Block Number 11.	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release:

Distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

none

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Restructurer, Version IIA Release 2 Data Translator, Reader, Writer, Translation Definition Language (TDL), TDL Analyzer, IDS Analyzer, Database, Records, Items, Groups.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Version IIA Release 2 Data Translator User Manual is intended to provide a complete guide for a WWMCCS H-6000 or 6000 user in the use of the University of Michigan software built to restructure IDS databases. The manual contains an overview of the restructuring process, guides in writing the necessary Data Translator specifications, control card set-ups, examples of output, and error message explanations.

409 349

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

COMMAND AND CONTROL TECHNICAL CENTER

Computer System Manual CSM UM 236-77

March 1977

DATA TRANSLATOR VERSION IIA RELEASE 2 USER MANUAL (Revision 1)

BY

ERIC KINTZER, JAMES BODWIN, WILLIAM COOL,
LINDA HUTCHINS, ANDREW MARINE, KENNETH MOORE,
DONALD SWARTWOUT, GREG WOLFE

DT 3.4

MARCH 1977

CLEARED
FOR OPEN PUBLICATION

JUL 13 1977 21

DIRECTORATE FOR FREEDOM OF INFORMATION
AND SECURITY REVIEW (OASD-PA)
DEPARTMENT OF DEFENSE

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314

DATA TRANSLATION PROJECT
UNIVERSITY OF MICHIGAN
Graduate School of Business Administration
Ann Arbor, Michigan 48109
(313) 763-1100

Approved for public release;
distribution unlimited.

PREPARED FOR

Defense Communications Agency
CCTC/WAD
11440 Isaac Newton Square, N.
Reston, Virginia 22090

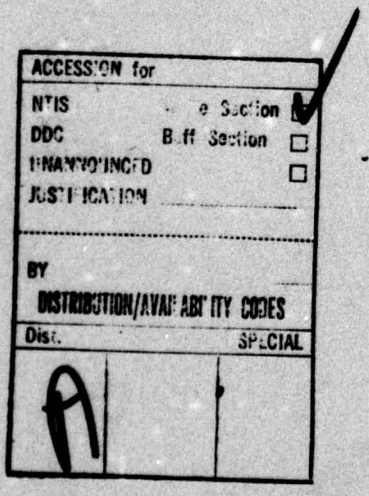
UNCLASSIFIED

Data Translation
Working Paper DT-3.4

DATA TRANSLATOR
VERSION IIA RELEASE 2
USER MANUAL

Data Translation Project
276 Business Administration
University of Michigan
Ann Arbor, Michigan
48109

December 1976
Working Paper



Prepared for
Defense Communications Agency
11440 Isaac Newton Square, N.
Reston, Virginia 22090
Contract DCA 100-75-C-0064

This document has not been cleared for open publication.

PREFACE

This User Manual provides detailed instructions for the preparation and definition of databases to be processed by Version IIA Release 2 Data Translator. The Version IIA Release 2 Translator is the third prototype implementation of a generalized data conversion and restructuring capability developed by the University of Michigan for CCTC/WAD. It is the first "operational" translator and therefore should be considered experimental in nature. Our purpose in releasing the translator and associated Users Manual is to allow the User Community a chance to use and comment on the desirability of this capability and documentation. Such feedback would provide the basis for improved documentation and software capability.

It is assumed that the reader has a reasonably high level of Honeywell expertise. Specifically, the user should be familiar with:

Sequential, ISP, and IDS files

Control card sequences

IDS database design

Timesharing

Please address all comments to both -

Mr. Paul Rubin, C420
Defense Communications Agency
11440 Isaac Newton Square, N.
Reston, Virginia 22090

Mr. James P. Fry
Data Translation Project
Graduate School of Business Administration
University of Michigan
Ann Arbor, Michigan 48109

(313) 763-1100
(703) 471-4227
(703) 471-1024

TABLE OF CONTENTS

TABLE OF CONTENTS

		Page
1.0	Introduction to Data Translation	1-1
	1.1 Background to Translation	1-1
	1.2 Capabilities and Limitations of the Version IIA Release 2 Data Translator	1-1
	1.3 Overview of Translation	1-5
	1.3.1 Preliminary Procedures	1-6
	1.3.2 The Data Translation Process	1-7
	1.3.3 Database Description	1-7
	1.3.4 Data Translation Modules	1-10
	1.3.4.1 The Reader	1-10
	1.3.4.2 The Restructurer	1-10
	1.3.4.3 The Writer	1-12
	1.4 Definition of Terms	1-12
	1.5 Resource Requirements	1-21
	1.5.1 Hardware Configuration	1-21
	1.6 Translation Summary	1-21
2.0	Preparing for Data Translation	2-1
	2.1 RIF Constructs	2-1
	2.2 The Restructurer's Working View of Databases	2-3
	2.2.1 Set-significant Items	2-3
	2.2.2 Augmented Bachman Diagrams	2-5
	2.3 Primary Keys	2-5
	2.4 Identifying Restructuring Transformations	2-10
	2.4.1 Step 1 - Materials Required	2-10
	2.4.2 Step 2 - Semantic Validity of Proposed Restructuring	2-13
	2.4.3 Step 3 - Feasibility of Proposed Restructuring	2-15
	2.5 Sequence of Translator Steps	2-18
3.0	Augmenting the IDS MD with Level 61 Information	3-1
	3.1 Process of Extending the IDS MD Section	3-1
	3.1.1 Further Rationale for Level 61 Entries	3-5
	3.1.2 Summary of Required Extensions	3-5
	3.2 Level 61 Rules	3-5
	3.2.1 Global MD Section Rules	3-6
	3.2.2 Global Level 61 Rules	3-9
	3.2.3 IDS Analyzer Automatically-generated Names	3-9
	3.2.4 Complete Level 61 Syntax	3-14
	3.3 Group Definition	3-16
	3.4 Item Definition	3-22
	3.5 Primary Key Definition	3-24
	3.6 Relation Definition	3-31
	3.6.1 Match-key Relations	3-31
	3.6.2 Phantom Pointer Relations	3-37
	3.7 ISP and Sequential Databases	3-43
	3.7.1 Identifier Items for ISP and Sequential Databases	3-45
	3.8 Multiple Source Databases	3-47

		Page
4.0	Writing TDL	4-1
	4.1 Basic TDL - Tree Transcription	4-1
	4.1.1 Preliminaries - Comments, Toggles, Macros	4-1
	4.1.2 TDL Structure	4-2
	4.1.3 TARGET RECORD Statement	4-3
	4.1.4 TDLAP Statement	4-3
	4.1.5 SOURCE RECORD Statement	4-6
	4.1.6 Item Assignments	4-6
	4.1.7 Selection Criteria	4-9
	4.1.8 Item Assignments and Comparisons Together	4-10
	4.1.9 Additional SOURCE RECORD Features	4-10
	4.1.10 Item vs. Item Comparisons	4-24
	4.1.11 Multiple Assignments and Comparisons	4-24
	4.2 Advanced TDL Features	4-25
	4.2.1 ACCEPT/REJECT IF NULL	4-25
	4.2.2 Use of Set-significant Source Items	4-29
	4.2.3 User-Supplied Qualification and Conversion Routines	4-30
	4.3 TDL Syntax Summary	4-35
5.0	Running the IDS Analyzer	5-1
	5.1 Detailed Overview of IDS Analyzer Execution	5-3
	5.2 Explanation of Processing Flow	5-5
	5.2.1 Activity 1 - Initialize Query Dictionary	5-6
	5.2.2 Activity 2 - Create IDS Query Dictionary	5-6
	5.2.3 Activity 3 - Execute IDS Analyzer Phase 1	5-7
	5.2.4 Activity 4 - Execute IDS Analyzer Phase 2	5-11
	5.3 IDS Analyzer JCL	5-11
	5.4 IDS Analyzer - Activity 1	5-13
	5.5 IDS Analyzer - Activity 2	5-15
	5.6 IDS Analyzer - Activity 3	5-19
	5.7 IDS Analyzer - Activity 4	5-22
6.0	Running the TDL Analyzer	6-1
	6.1 Detailed Overview of TDL Analyzer Execution	6-1
	6.2 Explanation of Processing Flows	6-3
	6.3 TDL Analyzer JCL	6-3
	6.4 TDL Analyzer - Activity 1	6-6
	6.5 TDL Analyzer - Activity 2	6-8
	6.6 TDL Analyzer - Activity 3	6-9
	6.7 TDL Analyzer - Activity 4	6-12
	6.8 TDL Analyzer - Activity 5	6-14
7.0	The Reader Module	7-1
	7.1 Reader Submodules	7-1
	7.2 Reader Processing Flow	7-1
	7.2.1 Restrictions and Warnings	7-6
	7.3 Reader JCL	7-6
	7.3.1 IDS Reader JCL	7-6
	7.3.2 ISP Reader JCL	7-11
	7.3.3 WWDMS Sequential Reader JCL	7-12
	7.4 Files and Reports	7-12

7.5 IDS Reader Files and Reports 7-13
7.5.1 Activity 1 7-13
7.5.2 Activity 2 7-13
7.5.3 Activity 3 7-14
7.5.4 IDS Reader Output and DRT Dump 7-16
7.5.5 Source RIF Dump 7-21
7.5.6 Debug Output 7-21
7.6 ISP Reader Files and Reports 7-21
7.6.1 Activity 1 7-22
7.6.2 Activity 2 7-22
7.6.3 ISP Reader Output 7-23
7.7 Sequential Reader Files and Reports 7-25
7.7.1 Activity 1 7-25
7.7.2 Activity 2 7-25
7.7.3 Sequential Reader Output 7-26

8.0 Running the Restructurer 8-1
8.1 Major Components of the Restructurer 8-1
8.2 Restructurer Processing Flow 8-3
8.3 Restructurer Control Card Deck Setup 8-3
8.4 Activity 01 Control Cards 8-6
8.4.1 File Code Assignments 8-6
8.4.2 User Parameters 8-8
8.4.3 Output Interpretation and Example 8-8
8.4.4 Errors 8-8
8.5 Activity 02 Control Cards 8-9
8.5.1 File Code Assignments 8-9
8.5.2 User Parameters 8-9
8.5.2.1 The $LIMITS Card 8-9
8.5.2.2 User Input Directives 8-11
8.5.2.3 Optional #REMOTEs and $PRMFLs 8-15
8.5.3 Output Interpretation and Example 8-15
8.5.4 Errors 8-22
8.6 Activity 03 Control Cards 8-22
8.6.1 File Code Assignments 8-22
8.6.2 User Parameters 8-22
8.6.3 Output Interpretation and Example 8-22
8.6.4 Errors 8-22
8.7 Modifications to Restructurer Control Cards for User Routines 8-24

9.0 Running the Writer 9-1
9.1 Detailed Overview of Writer Execution 9-2
9.2 Explanation of Processing Flow 9-5
9.2.1 Activity 1 - Utility Copy of Target RIF 9-5
9.2.2 Activity 2 - Initialize IDS database 9-5
9.2.3 Activity 3 - Compile COBOL-Writer Main Program 9-6
9.2.4 Activity 4 - Execute the Writer 9-6
9.2.5 Processing Limitations on Writer 9-10

9.3 Complete JCL to Execute the Writers 9-10
9.4 Activity 1 - Utility Copy of Target RIF 9-13
9.5 Activity 2 - Initialize Target IDS Database 9-14
9.6 Activity 3 - Compile COBOL-Writer Main Program 9-14
9.7 Activity 4 - Execute the Writer 9-16
9.8 Activity 5 - Copy Updated RIF Back into Permanent File 9-24

10.1 Validating Output 10-1
10.1 COBOL Application Programs 10-1
10.2 IDS Database Dumps 10-1
10.3 WWDMS Queries 10-1

11.0 Example Translation 11-1

Appendix A - IDS Analyzer Errors
Appendix B - TDL Analyzer Errors
Appendix C - Reader Errors
Appendix D - Restructurer Error Messages
Appendix E - Writer Errors
Appendix F
F.0 ADBMS Overview F-1
F.1 Describing an ADBMS Database -- The DDL F-1
F.2 The DDL Analyzer/Database Initializer F-2
F.3 The ADBMS Database and Database Tables F-2
F.4 Multiple Databases F-2
F.5 Database Keys F-2
F.6 The Work Database F-4
F.7 Restrictions on ADBMS F-4
F.8 Error Messages F-4
F.9 References F-6
Appendix G System Generation

1.0 INTRODUCTION TO DATA TRANSLATION

The Data Translation Project at the University of Michigan has developed a Data Translator which is capable of reorganizing WWDMS databases (Sequential, ISP, and IDS) by altering the logical structure and by modifying the physical structure. The work was completed on a Honeywell 6060 computer under a contract from the Command and Control Technical Center WWMCCS ADP Directorate (Code 400) of the Defense Communications Agency. The Data Translator is a complex and sophisticated software package, yet is understandable and easy to use. This manual describes the capabilities of the Data Translator, how to use the software, and an example of its use.

1.1 Background to Translation

This Data Translator is the product of many years of research and development. Previous Translators lacked many of the features of the current release. Refer to Table 1-1 for a comparison of Data Translator features. The Version I was developed to show the feasibility of the generalized Data Translator. Version II was to be an extension of Version I since it was capable of handling tree-type data structures. The Version IIA Release 1 was designed in response to the demand for a network Data Translator. The Release 2 overcomes Release 1's performance problems and presents some new features. All of the capabilities of the Release 2 Data Translator are detailed in Section 1.2.

Originally, when a user needed to represent his data in a new way, he would write, or have written, a program to change the data (Figure 1-1). If a different data representation were required, another program was needed. The cost of data translation was, therefore, very high since a large amount of programming effort was used in a one time application.

The current state-of-the-art approach to data translation is to use a generalized data translator (Figure 1-2). This approach is exemplified by the Version IIA Release 2 Data Translator. The generalized approach uses programs which have been written and descriptions of the old and new databases (herein referred to as source and target databases) and of the transformation between source and target databases. The architecture of the Data Translator will be described in Section 1.3.

1.2 Capabilities and Limitations of the Version IIA Release 2 Data Translator

1. Up to five source databases can be combined and restructured. The source databases may comprise any combination of WWDMS sequential, ISP or IDS databases.
2. Up to five target databases can be written. The target databases must be legal IDS databases.
3. The physical structure of the target database may be different from that of the source database. Therefore, page range, page size, PLACE NEAR, etc., can be changed to improve database performance.
4. User-implemented relations, such as phantom chains, pointer arrays, or match-keys, within and between source databases can be restructured into legal IDS chains in the target database.

Data Translator	Date Completed	Source DBMS	Target DBMS	Restructuring	Language Analyzers †	Typical CPU Time *	Internal Data Model
Version I	1973	NIPS, WWDMS Sequential	NIPS, WWDMS Sequential	Some	SDDL	-----	
Version II	Not Implemented	-----	-----	more	-----	-----	Hierarchical
Version IIA Release 1	June 1976	IDS	IDS	Very powerful	SDDL IDS MD TDL	80 years	Network
Version IIA Release 2	December 1976	WWDMS Sequential, ISP, IDS	IDS	Complete	IDS MD TDL	1 weekend	Network and Relational

Table 1-1: Feature Comparison

* CPU Time required to translate a typical database containing approximately 50 record types and 150,000 record instances

† SDDL - Stored Data Definition Language
TDL - Translation Definition Language

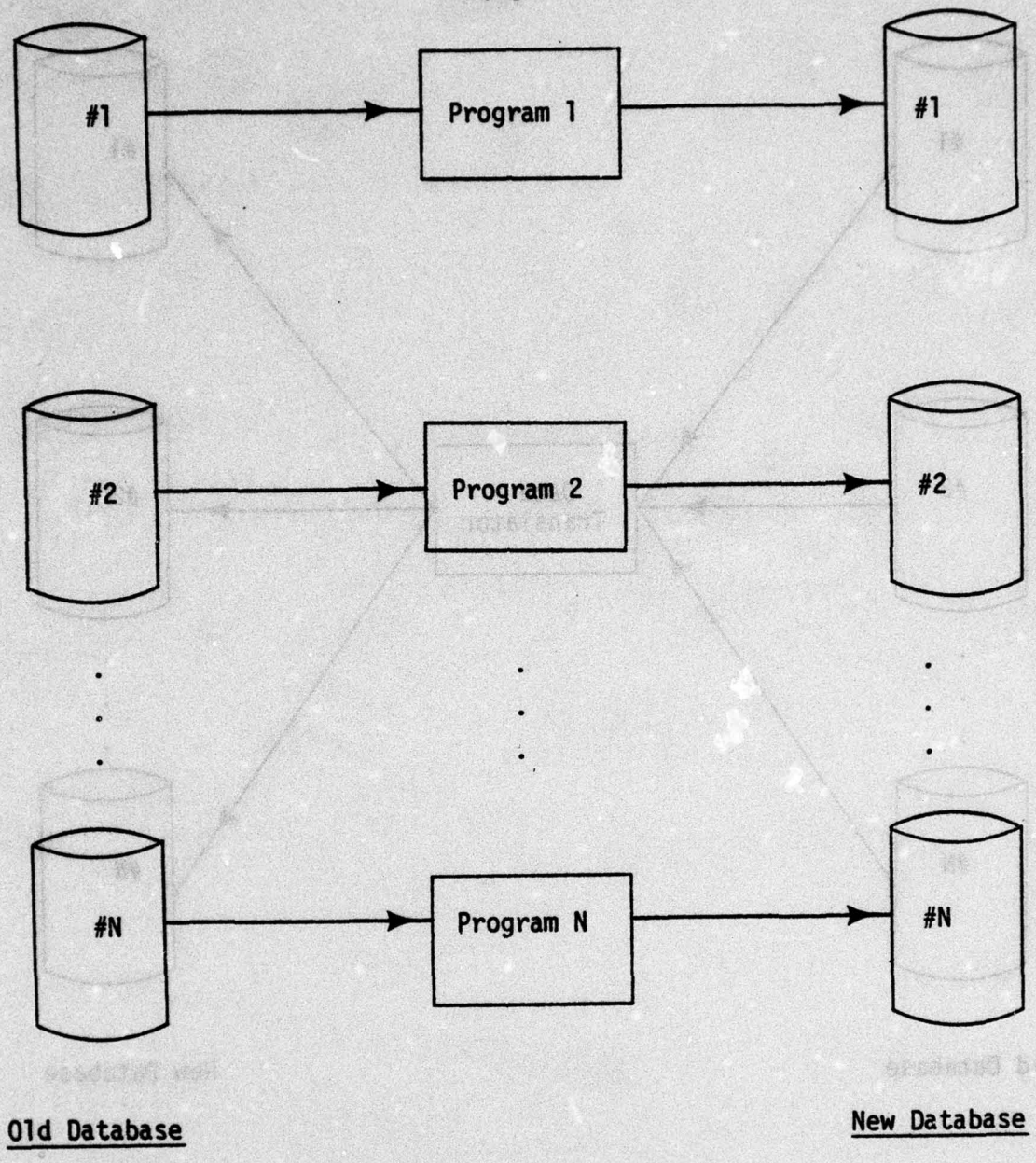


Figure 1-1
Conversion Program Approach to Data Translation

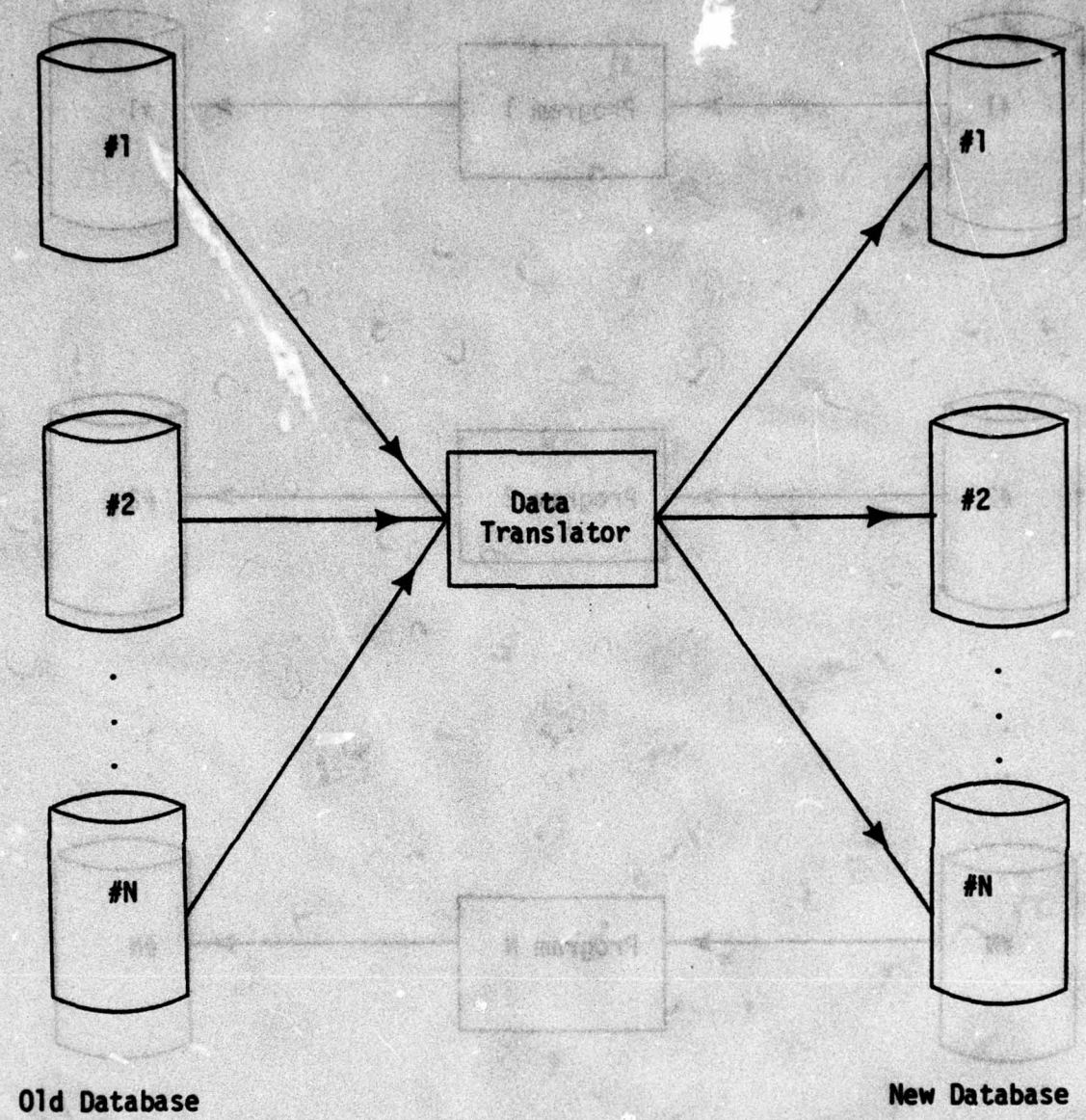


Figure 1-2
Data Translator Approach to Data Translation

5. Contained-in-repeating groups at the 03-49 level in the source database can be expanded into their own 01 level record types to become the detail of the 01 record in which they were contained.
6. Data items can be added, deleted or have their physical representation altered between the source and target databases.
7. Relations between records can be added, deleted or modified between the source and target databases depending on user-specified criteria.
8. Records can be added, deleted, or have their item contents changed between the source and target databases, depending on user-specified criteria.
9. New record types can be created in the target database from aggregations of source records. Duplicate data can be eliminated or created.
10. To facilitate the translation of large databases, the translation process can be stopped and restarted at a later time.

1.3 Overview of Translation

There are many reasons why a user would desire a new database structure. Common reasons include the need to satisfy new processing requirements, and performance improvement. For example, suppose a company had the following ISP data structure relating divisions in the company, its products, suppliers and parts:

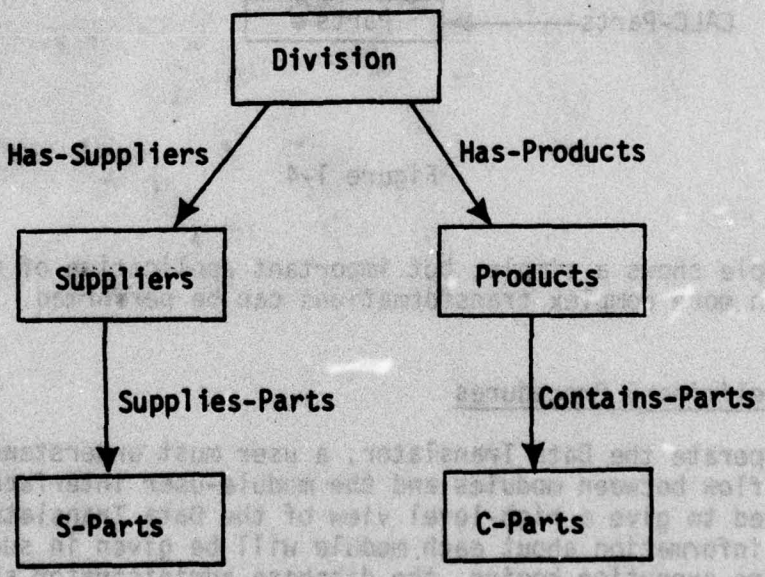


Figure 1-3
Source Database

This implementation of the database was initially successful, but since the company has grown and increased its product line, the old database is no longer sufficient. This inadequacy stems from the following reasons:

1. Updating parts records takes a long time since parts data is stored in both S-PARTS and C-PARTS records.
2. Direct access of either parts record is slow due to extra time spent searching overflow chains.
3. Parts reports are hard to produce due to the duplicate part data.

The database administrator of the company has recognized these problems and decided that an IDS database with the following structure will perform better:

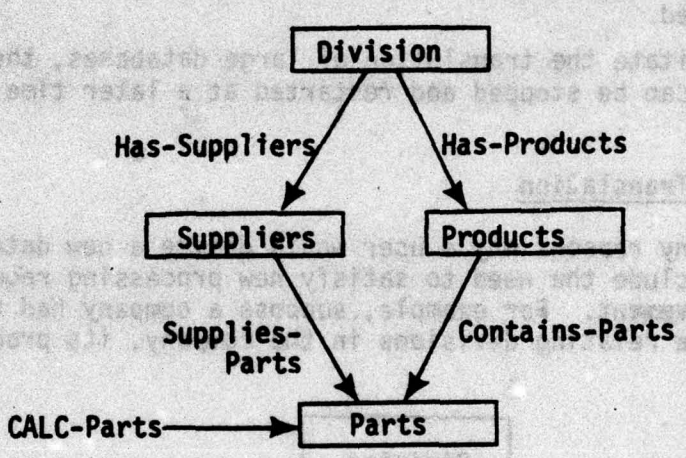


Figure 1-4

This example shows a simple, but important application of the Data Translator. Much more complex transformations can be performed.

1.3.1 Preliminary Procedures

To operate the Data Translator, a user must understand the connections and data flow between modules and the module-user interfaces. This section is intended to give a high-level view of the Data Translator modules. More specific information about each module will be given in succeeding sections.

Before execution begins, the database administrator should perform an analysis of the requirements for the target database. This involves drawing an IDS database diagram, writing an IDS MD section for the target, and determining how the records and chains in the target will be created from the source database.

Because the Data Translator allows the transformation of the user-implemented relations into IDS chains the database administrator should

examine the source database for these non-IDS constructs. If their information is to be preserved in the target, then explicit specification of the relation will be necessary in order for the Data Translator to be aware of their existence.

1.3.2 The Data Translation Process

To perform completely the data translation procedure, the user must complete six separate steps. The first two steps involve writing descriptions of the source and target databases using their MD sections and level 61 extensions. Those descriptions should be run through the IDS Analyzer to produce source and target SDDL (Stored Data Definition Language) tables. The third step is encoding the source-to-target transformation in the TDL (Translation Definition Language) and running the TDL Analyzer to produce TDL tables. The fourth step, Reader execution, creates the Restructurer Internal Form (RIF) of the source database(s). The Restructurer produces the target RIF database in the fifth step. The target IDS database(s) is produced by running the Writer. The entire Data Translation process is shown in Figure 1-5 and an example is included in Section 11.

1.3.3 Database Description

The Data Translator is a description-driven process. The translation modules must know the format of the source and target databases and the rules for creating records and chains in the target database from the records and chains in the source database.

The descriptions of the source and target databases are the source and target MD sections and additional information needed to restructure the database. If the source database is IDS, no new source MD section is necessary. If the source database is WWDMS Sequential or ISP, however, a IDS MD which describes the source database will have to be written. A MD section for the target database will also have to be written. After the MD sections have been collected or written, and additional information encoded as special level 61 statements, the extended MDs are ready to be used. The IDS Analyzer uses the extended MD sections to produce source and target SDDL tables. These first two steps are shown in Figure 1-6.

The SDDL tables are databases which hold information describing the source or target database. SDDL tables are analogous to the IDS Definition Structure which describes IDS databases. If multiple source or target databases have been described, there will be only one source or target SDDL tables file. Writing 61 levels is described in Section 3; running the IDS Analyzer is described in Section 5.

The final description to be written details the transformation between the source and target databases. That description must be written in the Translation Definition Language (TDL). The TDL describes how to create target records using the source records and chains. It is imperative that the TDL description be correct and validated. If the user does not write the TDL description properly, the output of the Translator will be invalid and the Restructuring will have to be repeated.

The third step of the Translation process is running the TDL description through the TDL Analyzer to produce TDL tables. The TDL tables cannot

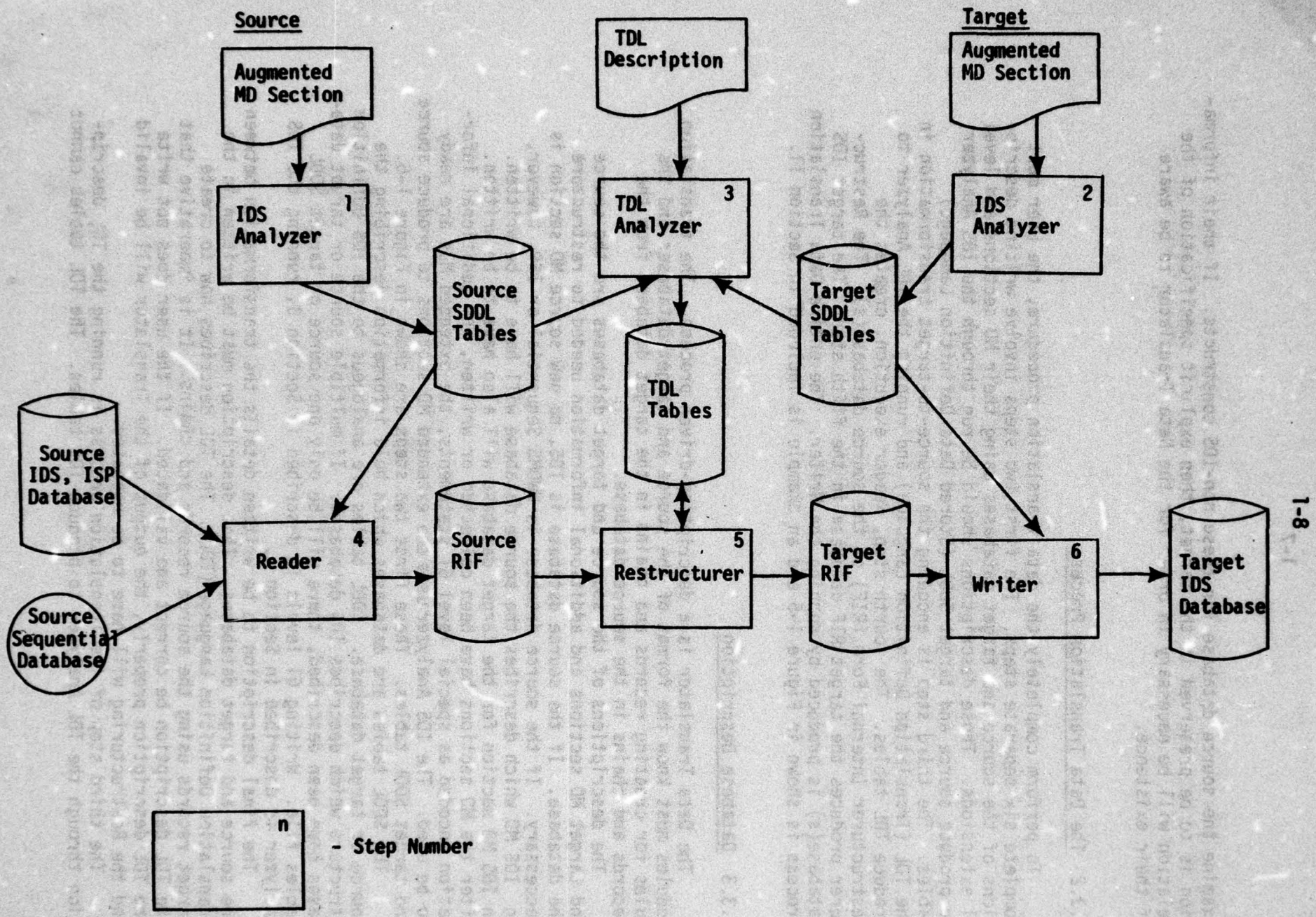


Figure 1-5
Version IIA Release 2 Architecture

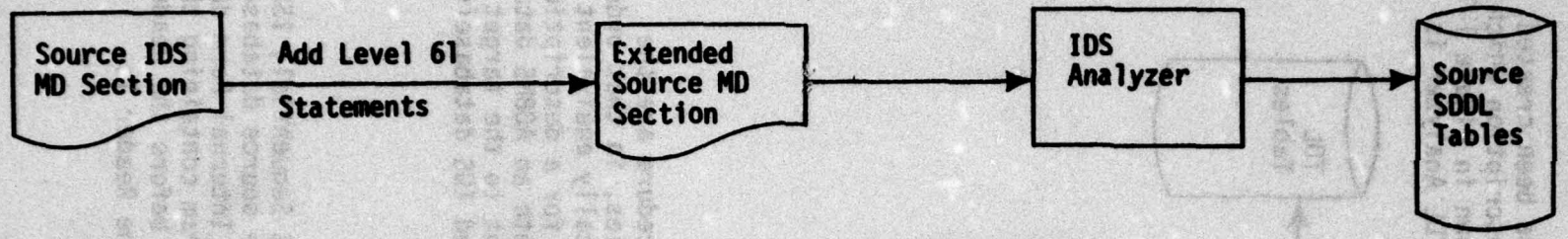


Figure 1-6a
Step One: Create Source SDDL Tables

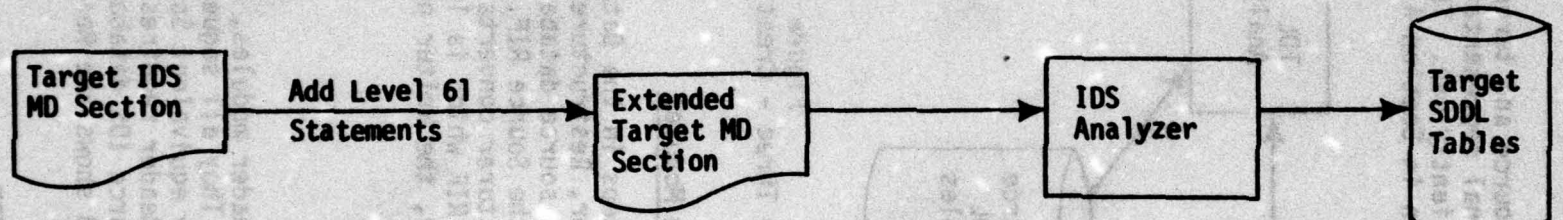


Figure 1-6b
Step Two: Create Target SDDL Tables

Figure 1-6
Steps One and Two of Data Translation

be created until both source and target SDDL tables have been created. The TDL tables are an internal representation of the TDL description which the Restructurer can conveniently use. TDL analysis is shown in Figure 1-7. Writing the TDL is shown in Section 4 and running the TDL Analyzer is described in Section 6.

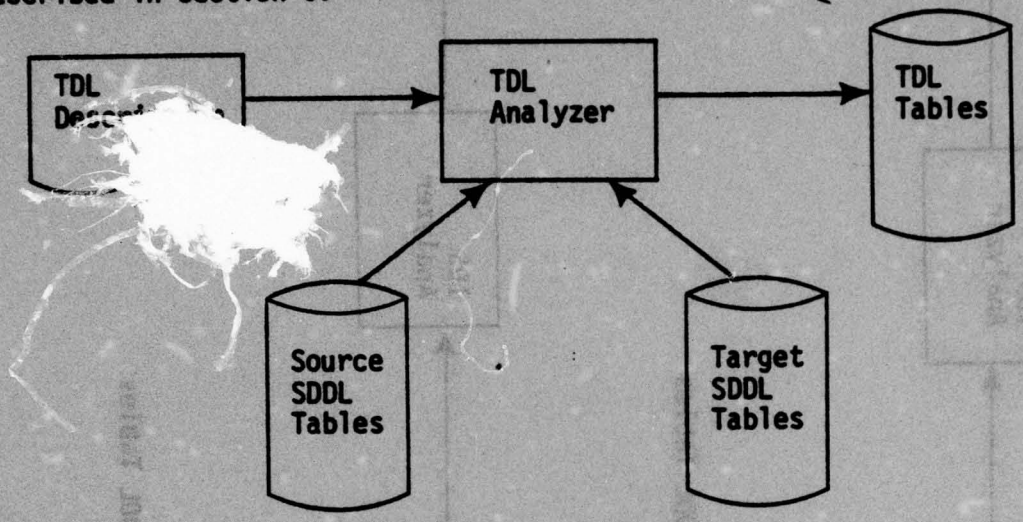


Figure 1-7
Step Three - Creating TDL Tables

1.3.4 Data Translation Modules

The final three steps in the Data Translation procedure are the executions of the Reader, Restructurer and Writer modules, in that order. The Reader converts the source database(s) into a logically equivalent ADBMS database called the Source RIF. (See Appendix F for a description of ADBMS.) The Restructurer converts the Source RIF into an ADBMS database called the Target RIF which is logically equivalent to the target IDS databases. Finally, the Writer produces the desired IDS database(s) from the Target RIF.

1.3.4.1 The Reader

There are three Reader modules, one each for WWDMS Sequential, ISP and IDS file systems. They all sequentially access the source database(s) and produce a logically equivalent Source Restructurer Internal Form database (SRIF). The IDS Reader requires a COBOL-IDS program containing the MD section for that Source IDS database to be compiled before the Reader can be run. Figure 1-8 shows the Reader process and the Reader is described in Section 7.

1.3.4.2 The Restructurer

The heart of the Data Translator is the Restructurer module. It uses information stored in the TDL tables to create a target RIF from the

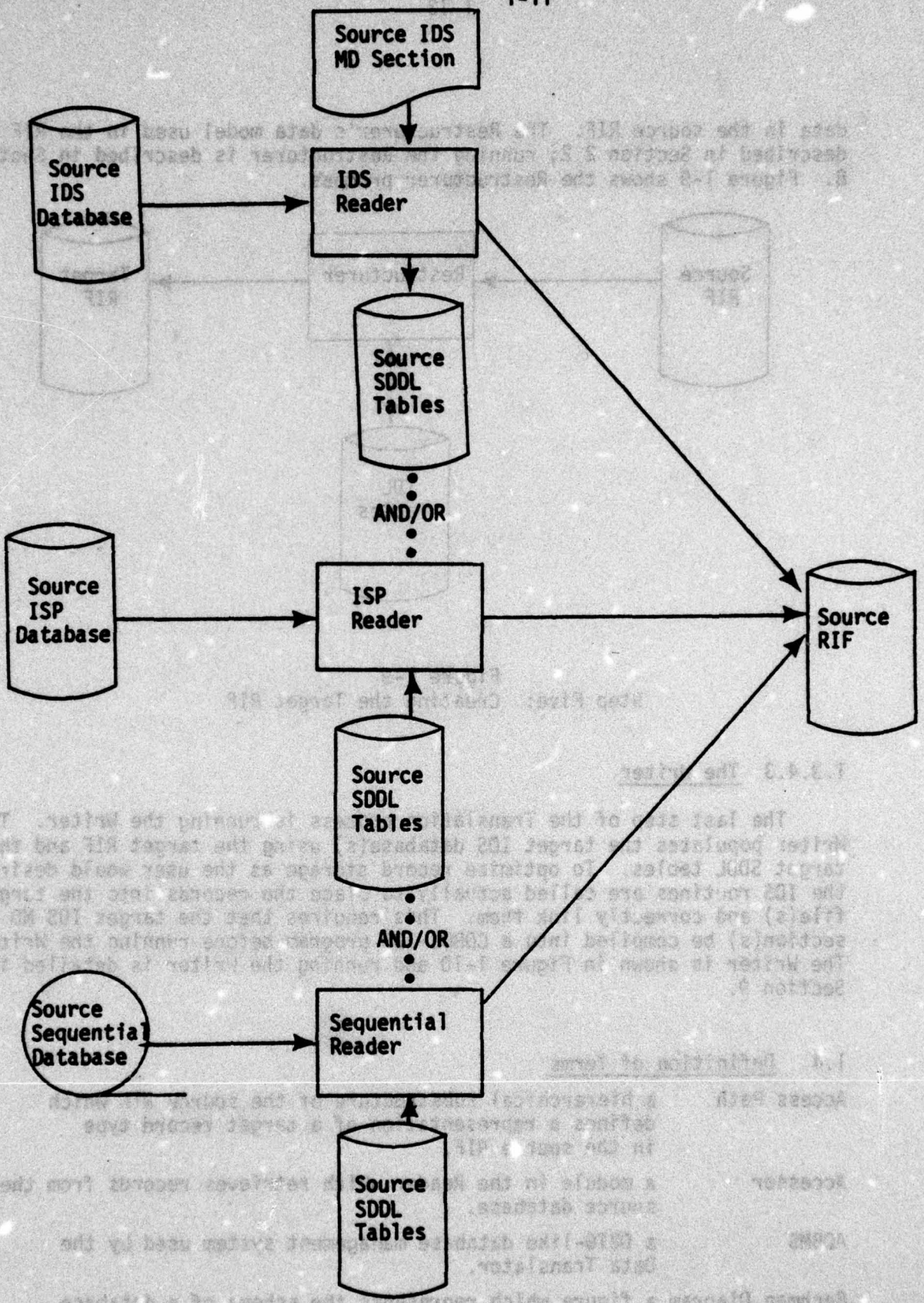


Figure 1-8
Step Four: Creating the Source RIF

data in the source RIF. The Restructurer's data model used in the RIF is described in Section 2.2; running the Restructurer is described in Section 8. Figure 1-9 shows the Restructurer process.

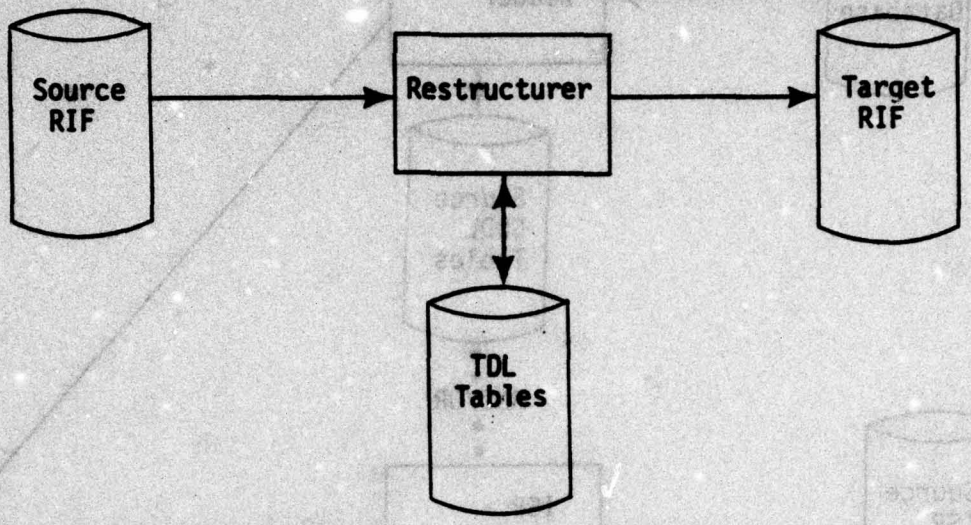


Figure 1-9
Step Five: Creating the Target RIF

1.3.4.3 The Writer

The last step of the Translation process is running the Writer. The Writer populates the target IDS database(s) using the target RIF and the target SDDL tables. To optimize record storage as the user would desire, the IDS routines are called actually to place the records into the target file(s) and correctly link them. This requires that the target IDS MD section(s) be compiled into a COBOL-IDS program before running the Writer. The Writer is shown in Figure 1-10 and running the Writer is detailed in Section 9.

1.4 Definition of Terms

Access Path	a hierarchical substructure of the source RIF which defines a representation of a target record type in the source RIF.
Accessor	a module in the Reader which retrieves records from the source database.
ADBMS	a DBTG-like database management system used by the Data Translator.
Bachman Diagram	a figure which represents the schema of a database. Record types are represented by boxes; set types are represented by lines connecting record type boxes.

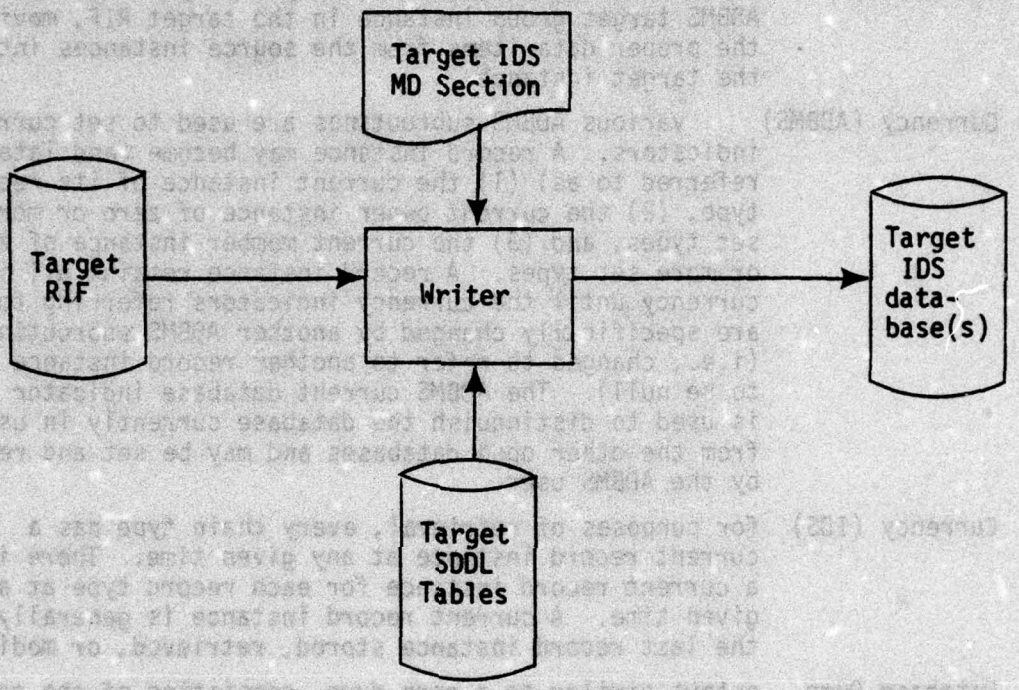


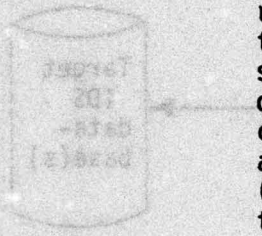
Figure 1-10
Step Six: Creating the Target IDS Database(s)

Chain (IDS) IDS's physical means of linking records together into relations. All chains are mapped as linked lists. The ADBMS equivalent is a set.

Contained-in-Repeating Group (CIRG) a named collection of items in a record. In COBOL-IDS, a CIRG is defined as any field with an OCCURS clause.

Constructor the Restructurer module that physically creates an ADBMS target group instance in the target RIF, moving the proper data items from the source instances into the target instance.

Currency (ADBMS) various ADBMS subroutines are used to set currency indicators. A record instance may become (and later be referred to as) (1) the current instance of its record type, (2) the current owner instance of zero or more set types, and (3) the current member instance of zero or more set types. A record instance retains all its currency until the currency indicators referring to it are specifically changed by another ADBMS subroutine (i.e., changed to refer to another record instance or to be null). The ADBMS current database indicator is used to distinguish the database currently in use from the other open databases and may be set and reset by the ADBMS user.



Currency (IDS) for purposes of retrieval, every chain type has a current record instance at any given time. There is also a current record instance for each record type at any given time. A current record instance is generally the last record instance stored, retrieved, or modified.

Database Dump output similar to a core dump, consisting of the contents of a database usually in both octal and character format. Each database page is dumped in its entirety (i.e., page headers, etc., are included) for debugging purposes.

Database Initializer (ADBMS) before a file can be used to hold an ADBMS database, it must be processed by the Database Initializer. The file is overwritten with zeroes, partitioned into pages, and each page is initialized with a page header record. The ADBMS DBT is written onto the file starting at page one. A record of the type SYSTEM is placed on the page following the DBT.

Database Initializer (IDS) all IDS databases, upon their creation, must be initialized with page header records before any data can be stored in them. This is done via utility routine QUTI.

Database Name the string of characters after the database declaration in the IDS Analyzer's run-time parameter file. This name is used by the Reader and Writer modules to establish currency in the SDDL tables.

Database Tables (DBT) the output of the DDL Analyzer. These tables contain a description of the database schema in a form usable by ADBMS and are stored in the first pages of the database.

Database Tables File (DBTF) a sequential file which contains the Database Tables for a database.

DDL - Data Definition Language a language used to describe the schema of a database in terms of the three basic ADBMS constructs: the item, the record, and the set. DDL text serves as input to the DDL Analyzer.

DDL Analyzer a language analyzer that accepts as inputs ADBMS DDL and produces as output Database Tables (DBT).

DDL Writer a module that scans SDDL tables and produces as output ADBMS DDL text suitable for input to the DDL Analyzer.

DDL Writer Work Database an ADBMS database used by the DDL Writer.

Detail every relation in IDS has one parent record type and up to 99 dependent record types. The dependent of a chain is called a detail. The ADBMS equivalent is a member.

DRT (Deferred Reference Table) an ADBMS database used by the populator to maintain information about subsets in the source RIF.

Extended MD Section *to describe databases to the Data Translator,* the source and target MD sections are augmented with additional information. The additional data is represented with level 61 entries.

Field (ADBMS) see item (ADBMS).

Field (IDS) IDS records are comprised of elementary items (fields) that represent a unit of data. IDS is aware of fields only at the 02 level.

Group see record (ADBMS).

Hash Database (ADBMS) an ADBMS database which contains one or more hash record types.

Hash Input (ADBMS) input required by the Database Initializer when initializing hash databases.

Hash Record (ADBMS) an ADBMS record type whose instances are stored and retrieved by hashing, i.e., calculating the record's address by randomizing on the primary key.

IDS (Integrated Data Store) a network database management system on the Honeywell H-6000 used as a source and the target of translation.

IDS Analyzer A Data Translator module that accepts an extended MD section, which describes a user's database in textual form and produces SDDL tables which are used by other translator modules.

IDS Query Dictionary — an IDS database produced from the extended MD section by the IDS Translator in query mode. It is read in as input to the IDS Analyzer.

IDS Structure Table — produced as the common area .IDS.. by every IDS compilation. It is an internal encoded table that describes an IDS database. It is formally known as the IDS Definition Structure.

IDS Translator — The IDS compiler invoked via the $IDS card. When configured in query mode, the IDS Translator produces the IDS Query dictionary.

Incremental Reading — the process of reading a source IDS database in multiple runs of the IDS Reader. The IDS Reader can be used to process the source database in a user-specified number of runs.

Instance — an occurrence of one object within a type or class, as distinguished from type. Example: an instance of record type PERSON is "JOHN SMITH".

ISP — an indexed-sequential file organization. Databases in ISP form (if conforming to WWDMS T-2 rules) can be converted to IDS databases by the Data Translator.

Item (ADBMS) — the elementary unit of data in an ADBMS record. Synonymous with field.

Item (IDS) — see Field (IDS).

Item Conversion — since only certain data types are supported by the Restructurer the Reader and Writer must perform item conversion for IDS items (fields) that are in data formats supported by IDS but not by the Restructurer.

Library — a file used to hold the object modules of several logically or functionally related subroutines.

Logically Deleted Record (ADBMS) — a record in an IDS database which has been flagged as deleted but has not yet been physically deleted from the database.

Master — every relation type in IDS has one parent record type and up to 99 detail record types. The parent of a chain is called a master. The ADBMS equivalent is owner.

Match-Key Relation — a non-IDS relation between record types in the source database(s). Implemented by having identical item values in different record instances.

Match-Key Set (ADBMS) — an ADBMS set implemented by storing the primary key of the owner record instance in the set-significant items of one or more member record instances.

Member — a dependent record type in an ADBMS set is called a member. The IDS equivalent is detail.

Multiple Database (ADBMS) refers to the fact that the Data Translator must have several databases open at the same time (SDDL tables, TDL tables, source and target RIFs, etc.). The multiple database capability is part of ADBMS.

Multiple Database (User) the Data Translator has the capability to accept as many as five source database schemas (IDS, ISP, or sequential or any combination thereof) and produce as many as five target IDS databases. Each target database can be derived from some or all of the source databases.

Owner a parent record type in an ADBMS set is called an owner. The IDS equivalent is master.

Page (ADBMS) an ADBMS database is made up of one or more database pages. The page is the smallest unit that can be moved from mass storage to core and back. The ADBMS page length is 1024 words.

Page (IDS) an IDS database is broken into physical access units called pages. Each page in a subfile is of identical size and must be a multiple of 64 words.

Partial Writing the process of outputting a target database over several execution runs. Permits the user to write a target file when computer time resources are constrained.

Phantom Pointer Relation a set or relation in an IDS database which is maintained by the user. Set membership is based on IDS reference codes stored in user data items.

Physically Deleted Record (ADBMS) see Logically Deleted Record (ADBMS),

Physically Deleted Record (IDS) a record which was logically deleted from an IDS database, has been removed from all of its sets and its space on the page reclaimed.

Plex Group a group or record in a network database that is a member of more than one set type (ADBMS) or chain type (IDS).

Pointer Array a CIRG in an IDS database in which the user has stored IDS reference codes. (see also Phantom Pointer).

Populator a module in the Reader which controls the set linking process in the source RIF.

Primary Key (ADBMS) a collection of data items in an ADBMS hash record type whose combined values uniquely determine an instance of the hash record type and are used in the storage and retrieval process.

Qualification in the restructuring process, the testing of source item values to determine if and/or how a target group instance is to be built.

Qualifier the Restructurer module that performs qualification.

Query a request to a database for information that satisfies one or more specified criteria. The request is made via the Database Management System.

Reader	the Translator module which transforms the user's database(s) into a logically equivalent ADBMS database (source RIF). There are three Reader modules: one each for IDS, ISP, and Sequential files.
Real Parent	the actual owner instance of a set instance in the source RIF. (see also Surrogate Parent).
Record (ADBMS)	a named collection of zero or more items. Synonymous with group.
Record (IDS)	a named collection of zero or more fields.
Reference Code	each record instance within an IDS database has a logical address which is unique. This address is known as a reference node and is divided into two parts - a page number and a line number within that page.
Relation	synonymous with set (ADBMS) and chain (IDS).
Restructurer	the Translator module that produces the target RIF from the source RIF and the TDL tables.
Restructuring	the process of altering the logical structure or schema of a database.
Restructuring-by-Parts	the process of restructuring a database using several Restructurer runs as opposed to a single run. Each run processes only a few access paths at a time. The target records produced by each run are stored in the same target RIF.
RIF (Restructurer Internal Form)	the standardized database format provided by ADBMS along with certain restrictions required by the Version IIA Release 2 restructuring algorithm.
Run-time Parameter	a data value, used as input to a program, which is specified in the JCL.
Run-time Parameter File	a data area from which run-time parameters are read.
Schema	a description of the logical structure of a database.
SDDL Tables	the output of the IDS Analyzer. The SDDL tables are an ADBMS database in which all the information contained in the extended MD section is stored.
Sequential Database	any user database not under the control of ISP or IDS, e.g., a system standard format file, manageable by GFREC. The Data Translator will accept sequential databases as input to the Reader provided that they conform to WWDMS T-2 restrictions.

Set an ADBMS construct used to define a relationship between two record types known as the owner and the member record types for the set type. An instance of an ADBMS ordered set is always implemented as a linked list consisting of one instance of the owner record type for that set type and one or more instances of the member record type for that set type. An instance of an ADBMS match-key set always consists of one instance of the owner record type and one or more instances of the member record type, each having the primary key of the owner record instance in their set-significant items. The IDS equivalent of the ADBMS ordered set is the chain.

Set-Significant Items (ADBMS) a collection of data items in the member record type of a match-key set. The items establish a member record instance in a match-key set instance when they contain the primary key of an owner record instance.

Sibling an IDS record type which is a detail of a chain type which also has other detail record types.

Source Accessor a Restructurer module that accesses record instances in the source RIF, driven by the stack that was passed to it by the Stack Builder.

Source Group an ADBMS record or group (either type or instance) in the source RIF. (Abbreviated SGROUP, SG).

SRIF (Source RIF) an ADBMS RIF database which is produced by the Reader from the user's source database(s).

Stack Builder a Restructurer module that reads access paths in tree format from the TDL tables and produces a stack which drives the Source Accessor.

Subfile a file which contains part of an IDS database. IDS databases can be divided into many GCOS physical files for a variety of reasons.

Subset a set instance in the source RIF which contains fewer record instances than the same set instance in the IDS database. Note that a set instance in the IDS database may be composed of many subsets.

Surrogate Parent a dummy record created in the source RIF by the Populator.

System Access Set in an RIF database, each record type is a member of a system-owned set type called its System Access Set. All instances of each record type are made members of their System Access Set for ease of retrieval.

System Entry Point those records in a database schema from which a database traversal may start. Examples are CALC records, or the top record in a tree structure. Furthermore, a relation is defined by the IDS Analyzer in which the system entry point record is the member and the SYSTEM record is the owner.

System Generation Tape — a tape which contains all of the files and JCL necessary to run the Data Translator.

SYSTEM Record — every ADBMS database has exactly one instance of record type SYSTEM placed in the database by the Database Initializer. The SYSTEM record instance allows the user to "enter" the database using ADBMS sets since it is always the current SYSTEM record.

Target Group (Target Record) — an ADBMS record (either type or instance) in the target RIF (Abbreviated TGROUP, TG).

Target Relation (Target SET) — an ADBMS set (either type or instance) in the target RIF.

TDL — Translation Definition Language, a language used to describe the mapping of the logical structure of the source RIF to the logical structure of the target RIF.

TDL Analyzer — a language analyzer that accepts as input TDL text and produces as output TDL tables.

TDL Tables — the output of the TDL Analyzer. The TDL tables are an ADBMS database in which all the information contained in the TDL text is stored.

Translation Library — a GCOS library file containing all of the ADBMS, GMAP and utility routines needed to run the Data Translator.

TRIF (Target RIF) — the ADBMS RIF database that is produced by the Restructurer from the source RIF and the specification in the TDL tables.

Type — a class of objects, as opposed to an instance or occurrence of an object of a type (e.g., a record type PERSON is composed of item types NAME, AGE, and SS NUM).

User Conversion Routine — a user-written routine that performs conversions on source RIF item values before they are assigned to items in the target RIF records. The existence of a converison routine is declared in the TDL description. The conversion routine is called by the Constructor when the target record instances are being built by the Restructurer.

User Input Directives — user-supplied textual statements that control certain aspects of a Restructurer run, e.g., whether or not debug output is to be produced.

User Input Processor — a Restructurer module that processes the User Input Directives.

User Qualification Routine — a user-written routine which performs a particular type of qualification. The existence of a qualification routine is noted in the TDL tables. It is called by the Qualifier when the target group instances are being built by the Restructurer.

Writer	the final translation module. Its job is to transfer data records from the target RIF into the target IDS database preserving all information content.

1.5 Resource Requirements

The Data Translator can be used only on Honeywell 6000 and 600 series computers. The following conditions must be satisfied to run the Data Translator.

1. IDS must be in use. ISP must be available to translate ISP databases.
2. One processor and at least 256K of core must be available.
3. Secondary storage (disk) at least three times larger than the size of the source or target databases should be available.
4. One 9-track tape drive should be available.
5. IDS Data Query must be available.

1.5.1 Hardware Configuration

It is difficult to determine an optimal system configuration for the Data Translator since each application and installation is unique. But, there are some rules to follow which will help to improve efficiency.

1. Each module has an estimated records/hour (CPU) rate. It is based on fifty to seventy-five characters per average record. Do not attempt to translate more data than is physically possible given existing computer time constraints. Make full use of the partial or incremental translation features described in Sections 7-9.
2. Attempt to perform translation on a dedicated system. Tests have shown that running multiple processors will statistically affect total elapsed time.

1.6 Translation Summary

Table 1-2 correlates the basic translation steps with the Input/Output requirements and the estimated time to perform each step.

Step	Description	Inputs	Outputs	Estimated Time to Perform:
1.	Write 61 levels for source database(s) and run IDS Analyzer	1. MD section and 61 levels	1. Source SDDL Table	1 man-day
2.	Write 61 levels and MD section to target database(s) and run IDS Analyzer	1. MD section and 61 levels	1. Target SDDL Table	2 man-days
3.	Write TDL Description and run TDL Analyzer	1. TDL Description	1. TDL Tables	2 man-days
4.	Run the Reader	1. Source IDS databases 2. Source SDDL tables 3. Source MD section	1. Source RIF	1-2 machine days
5.	Run the Restructurer	1. Source RIF 2. TDL tables 3. Target SDDL tables	1. Target RIF	1-2 machine days
6.	Run the Writer	1. Target RIF 2. Target SDDL tables 3. Target MD section	1. Target IDS database(s)	1 machine day

Table 1-2: Data Translation Summary

2.0 PREPARING FOR DATA TRANSLATION

This section describes the pre-translation preparations that must be completed before the translation specifications (i.e., 61 levels and TDL) can be written.

2.1 RIF Constructs

The Restructurer accepts only databases that conform to the rules for Restructurer Internal Form (RIF) databases. That is, the database must be a collection of records. Records are made up of items, and are related by means of sets ("set" is synonymous with the IDS term "chain". A set's owner type is the chain master, and the member type is the chain detail). Certain constructs are not permitted in a RIF database. If they exist in the source IDS database, their resolution into RIF constructs must be specified in 61 level statements added to the MD section. The use of 61 levels is described in detail in Section 3 of this manual.

Illegal constructs in RIFs:

1. Sets with multiple owner and/or member types. These must be resolved into several sets, one for each owner type/member type pair, as in Figure 2-1.

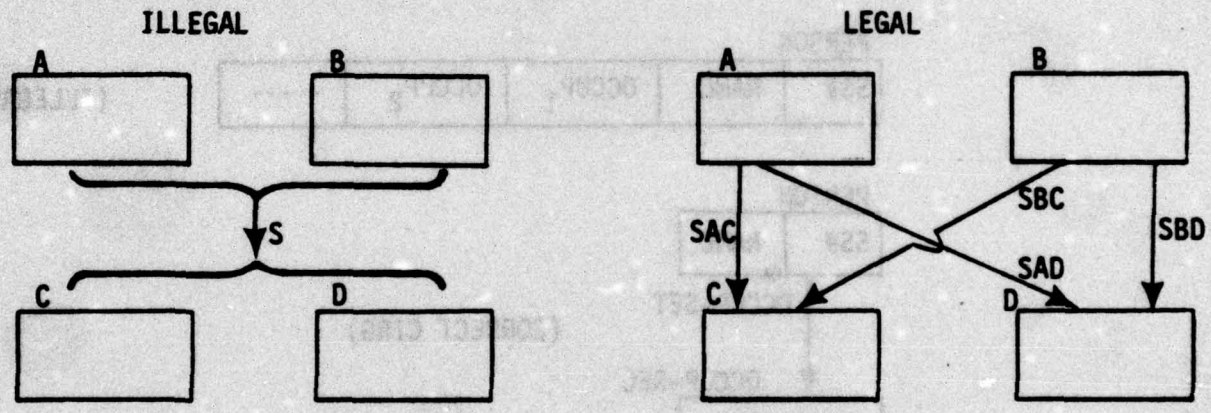


Figure 2-1
Resolution of Multiple Owner- and Member-Type Set

2. Phantom Chains. These are user-implemented chains whose pointers are IDS reference codes stored as fields in the master and detail records. Such chains can be declared in 61 level statements and implemented by the Reader as actual sets in the source RIF.

3. Naming groups. All elementary items in an MD section, regardless of level, are treated as simple items. For example, if the source MD section contains

```
01  HOUSE
    02  ADDR
        03  NUMBER
        03  STREET
        03  CITY
        03  STATE
        03  ZIP
    02  OWNER
```

then the source RIF HOUSE record would contain the items NUMBER, STREET, CITY, STATE, ZIP, and OWNER, but ADDR would have no meaning. If this is undesirable, an appropriate 61 level statement must be written.

4. Contained-in-repeating groups (CIRGs). Repeating groups and items must be expanded into new group types and new sets, as in Figure 2-2.

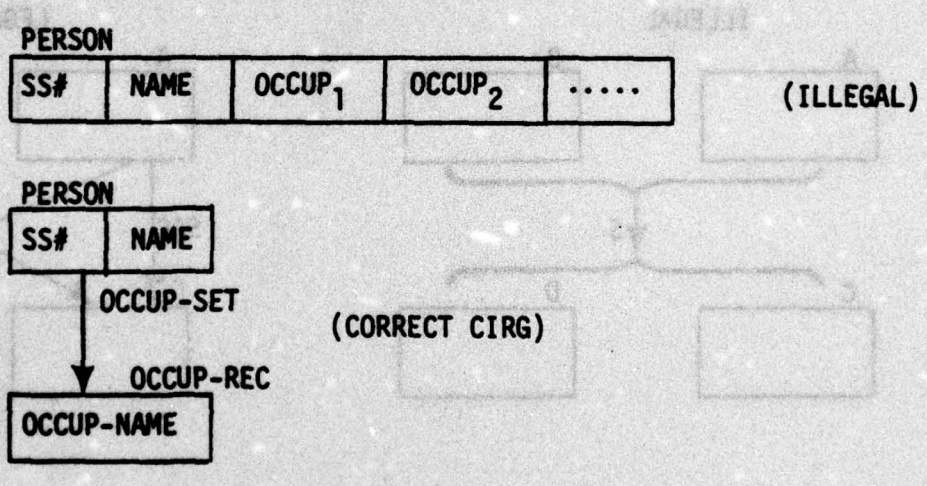


Figure 2-2
Resolution of a CIRG

5. Match-key sets. These are user-implemented sets in which one or more items in the owner and member record types are declared as match-key items. The match-key items in the member record type correspond one-for-one to the owner's match-key items. An instance of the match-key set consists of an instance of the owner record type and all the member record instances whose match-key item values are equal to the corresponding match-key item values in the owner record. Figure 2-3 gives an example of such a set. The STATE record is the owner type and the PEOPLE record, the member

type. The match-key set STATE-OF-RESIDENCE is implemented by the match-key items NAME in the STATE record and RESIDENCY in the PEOPLE record.

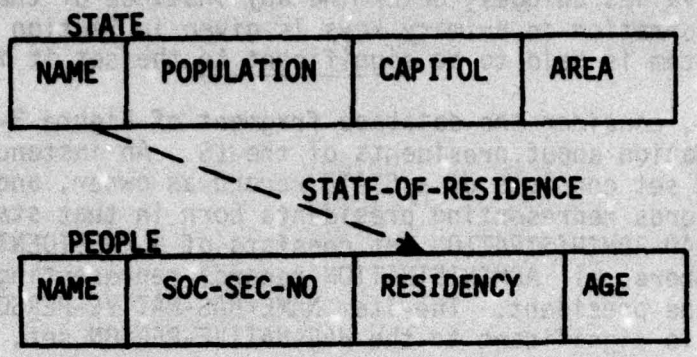


Figure 2-3
A Match-Key Set

Match-key sets are described by 61 level statements, and are constructed by the Reader in the source RIF as explicit sets.

2.2 The Restructurer's Working View of Databases

During the construction of a database, two important tasks must be accomplished:

a) all records must be constructed and given values for their data items, and

b) all sets must be constructed.

In particular, this applies to the construction of target RIF databases by the Restructurer. Task a) is performed in a straightforward way, as directed by the user's TDL descriptions. The Restructurer's approach to task b) is also straightforward, but requires that the user understand the concept of a set-significant item, which is described in the remainder of this section.

2.2.1 Set-significant Items

IDS, as well as all modules of the Data Translator, requires that if a record type is declared a member of a set which is system-owned, then every instance of the record type must be a member of the set. Thus no additional effort is required to establish system-owned sets. When a set is not system-owned, however, some means (e.g., pointer fields, pointer arrays, special data items) must be used to establish whether or not a given instance of the member record type belongs to the set, and if so, to identify the correct owner record instance.

During the restructuring phase of data translation, this is accomplished by storing in each target RIF instance of the member record type, a copy of the primary key of the corresponding owner record instance. If the record is not to belong to the set, a suitable null string is stored. This data

resides in a collection of set-significant items in the member record type. There is one set-significant item for each data item of the owner record type's primary key. The primary key of a record type is a collection of data items whose values uniquely determine any instance of the record type. More detailed information on primary keys is given in Section 2.3. A set-significant item is said to be significant to the set it is used to represent.

For instance, consider the database fragment of Figure 2-4 which represents information about presidents of the U.S. An instance of the HAS-NATIVE-PERSON set consists of a STATE record as owner, and as members, all PRESIDENT records representing presidents born in that state. An instance of the HAD-ADMINISTRATION set consists of a PRESIDENT record as owner, and as members, all ADMINISTRATION records representing administrations headed by the president. The item NAME<HAS-NATIVE-PERSON> in the PRESIDENT record is significant to the HAS-NATIVE-PERSON set, while the items LAST-NAME<HAD-AD>, FIRST-NAME<HAD-AD>, and INITIAL<HAD-AD> in the ADMINISTRATION record are significant to the HAD-ADMINISTRATION set. Primary key items are underlined.

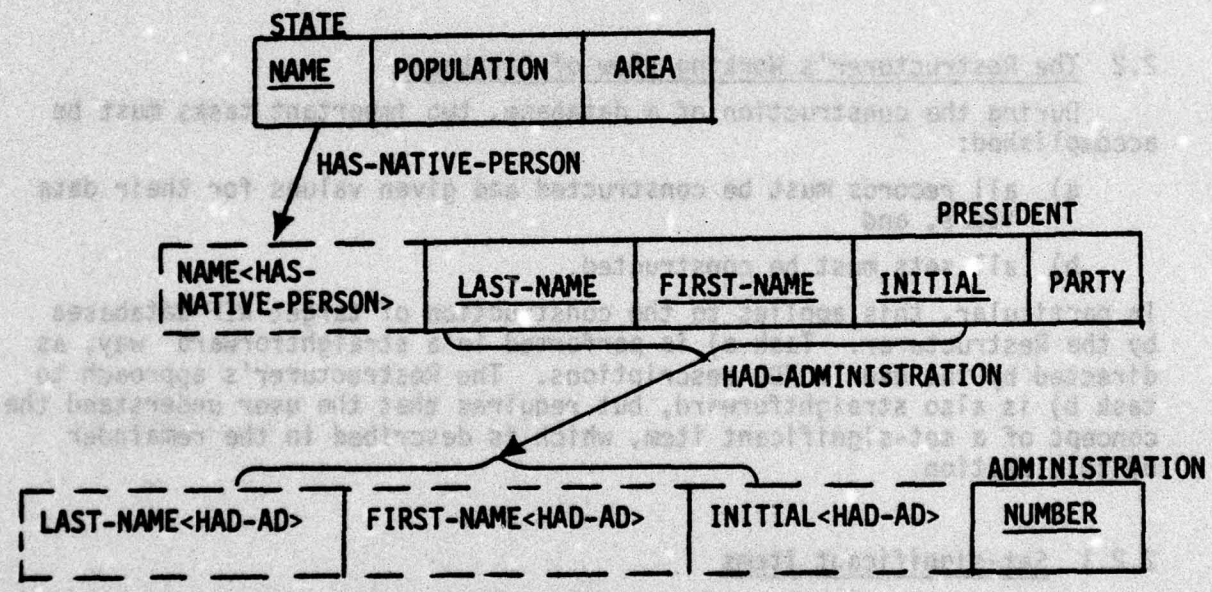


Figure 2-4
Set-significant Items

Set-significant items are identified and named by the IDS Analyzer, subject to the following rules:

1. Each set-significant item is significant to exactly one set.
2. The items significant to a particular set are in one-to-one correspondance with the items that make up the primary key of the owner record type.

3. Set-significant item names consist of the owner key item name followed by all or part of the set name enclosed in angle brackets.

Although set-significant items exist only in the target RIF, the IDS Analyzer identifies and names them for both source and target databases. This helps to insure uniform descriptions for user databases throughout the data translation process, and the source set-significant items may be used by the advanced TDL writer as if they existed in the source RIF.

2.2.2 Augmented Bachman Diagrams

At every stage of data translation, the user's chance of success is enhanced significantly by the use of carefully drawn Bachman diagrams to describe source and/or target data structures. Bachman diagrams used to describe source and target RIFs should be augmented to include all set-significant items, as well as to indicate which data items in a record make up its primary key, and which set a set-significant item is used to establish. This may be done as follows:

1. Augment the rectangles representing record types with a rectangle whose sides are dotted lines.
2. Fill the new rectangles with the set-significant item names generated by the IDS Analyzer. Keep the items significant to a particular set together and in the same order as in the primary key in the owner record type.
3. Draw arrows representing sets from the primary key item(s) of the owner record type to the set-significant item(s) in the member record type. When there are multiple items involved, horizontal braces at each end add clarity.
4. Underline each record's primary key.

Figure 2-4 is an example of an augmented Bachman diagram drawn in this way. It should be emphasized that augmented Bachman diagrams are useful whenever working with a source or target RIF, and they are an essential tool in writing complex TDL descriptions. They are also a convenient place in which to record set-significant item names after they are generated by the IDS Analyzer.

As a final example, Figure 2-5 shows an ordinary Bachman diagram in which all primary keys have been underlined, and the augmented Bachman diagram for the same schema.

2.3 Primary Keys

The Data Translator requires that a primary key be specified for each record type. A primary key of a record type is a collection of items whose combined values uniquely determine any instance of the record type. A record type may have many possible primary keys, but exactly one must be identified and used throughout the data translation process. For instance, consider the record shown in Figure 2-6. The item SOCIAL-SECURITY-NUMBER

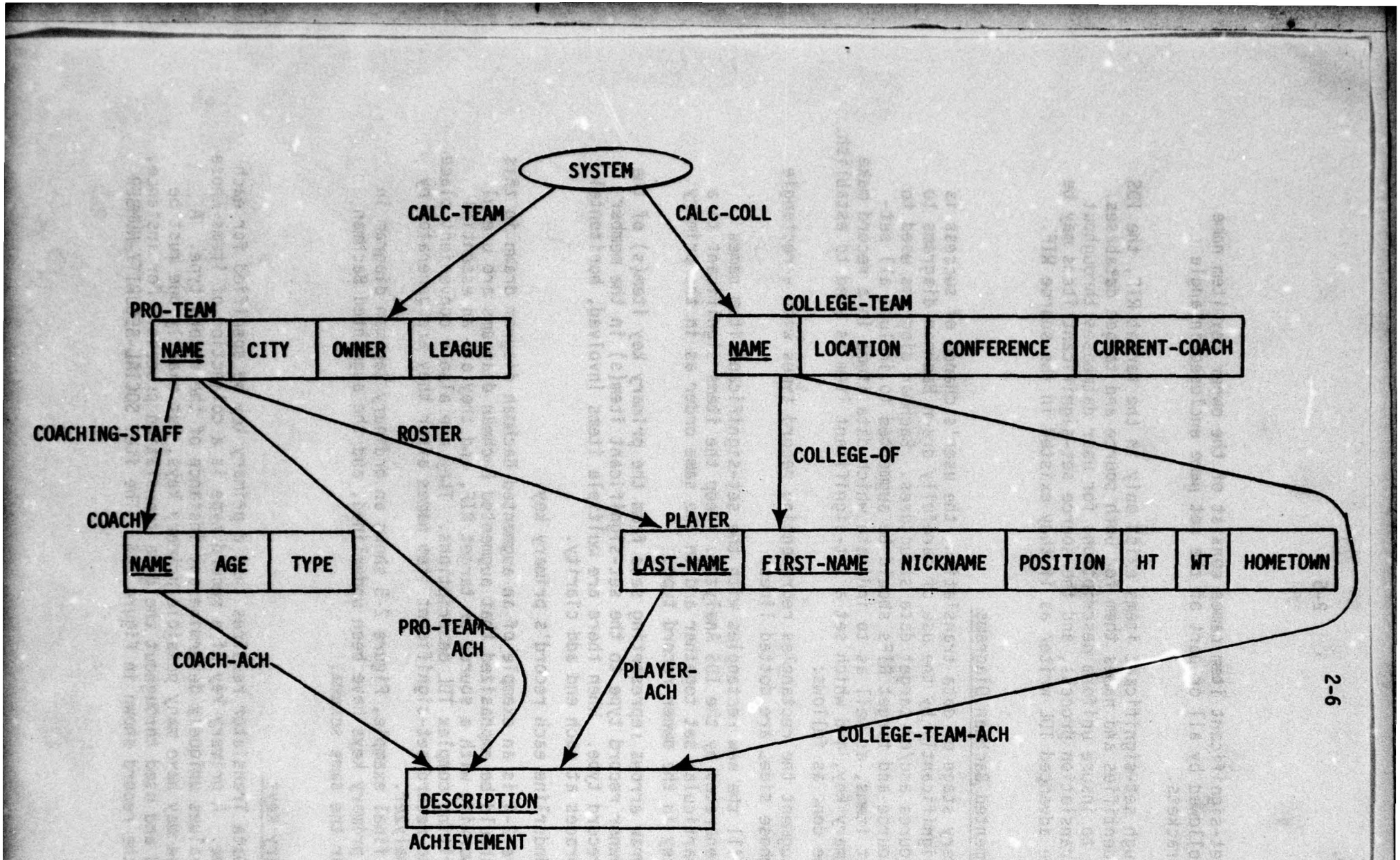


Figure 2-5
Football Database

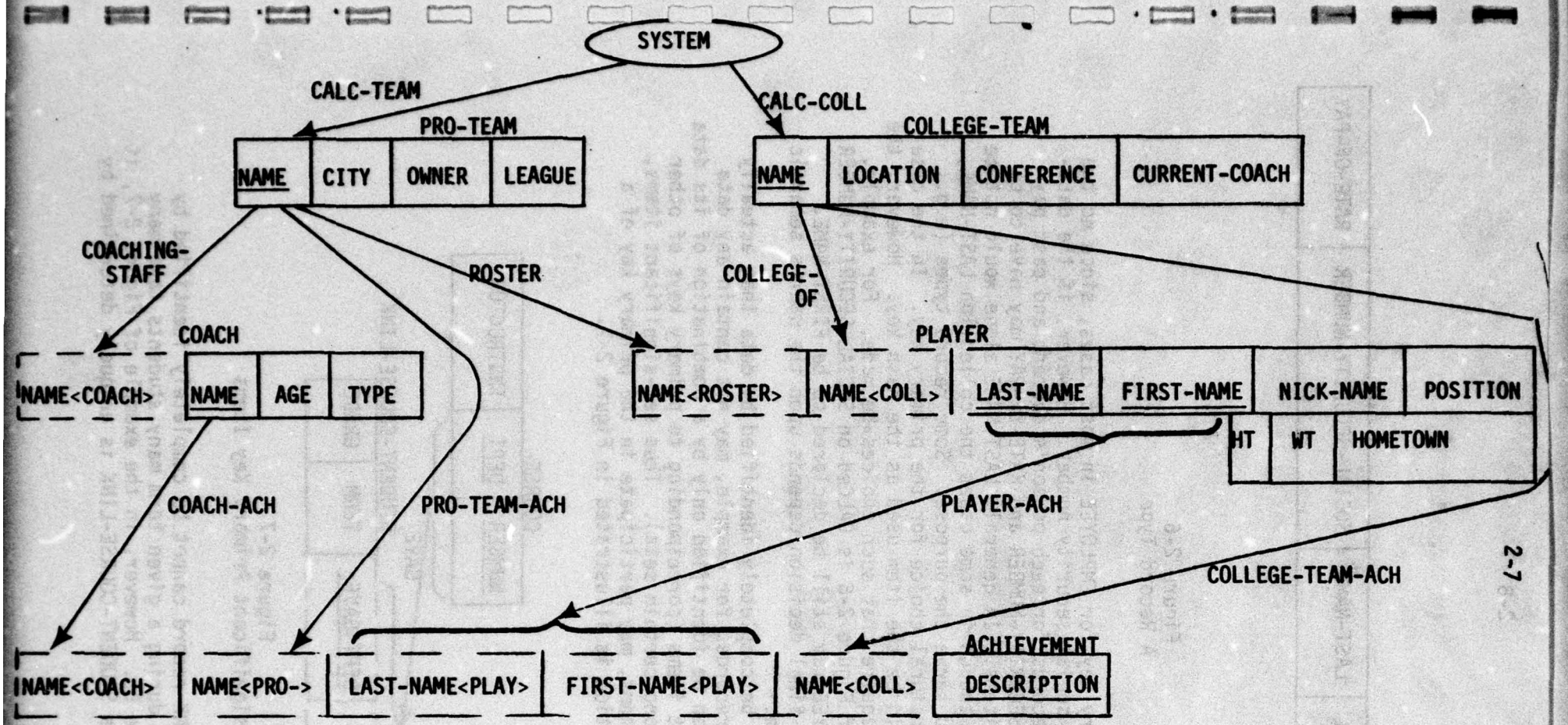


Figure 2-5 (cont'd)
Augmented Bachman Diagram for Football Database

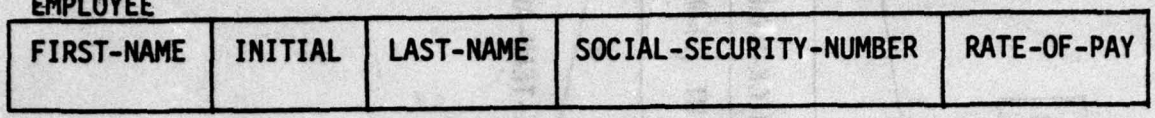


Figure 2-6
A Record Type

could be used as a primary key for EMPLOYEE in most cases, since no two people can have the same social security number. However, if the database contains EMPLOYEE records for each person's current and past pay grade, then both SOCIAL-SECURITY-NUMBER and RATE-OF-PAY may have to be declared as primary key items. In general, LAST-NAME alone would not be an adequate primary key, though in some cases the collection LAST-NAME, INITIAL, FIRST-NAME would serve the purpose. Some record types (e.g., CALC records) contain a natural choice for the primary key. In the case of a CALC record, it would be the item used as the hash key. However, the Data Translator cannot require that such choices be made. For example, if the EMPLOYEE record of Figure 2-6 is CALCed on SOCIAL-SECURITY-NUMBER, the primary key of EMPLOYEE may still be declared to be FIRST-NAME, INITIAL, LAST-NAME. The final decision depends upon the user's semantic knowledge of the database.

Some records cannot be completely identified by data that actually resides in them. (Link records, for example, may not contain any data items.) Such a record can be identified only by a combination of its data items and set-significant items (corresponding to primary keys of other records, which own it along certain sets). Thus set-significant items, as well as actual data items, may participate in the primary key of a record type. This situation is illustrated in Figure 2-7.

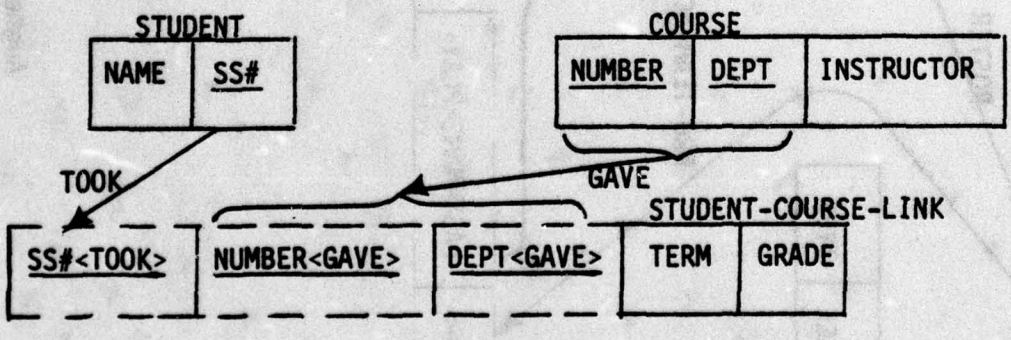


Figure 2-7
Set-significant Primary Key Items

A STUDENT-COURSE-LINK record cannot be completely identified by TERM and/or GRADE, since during a given term many students take many courses, and many receive Bs. However, in the example of Figure 2-7, it has been indicated that a STUDENT-COURSE-LINK is uniquely determined by

the SS# of the student who took the course, together with the course NUMBER and the DEPARTMENT that gave the course. If students are permitted to repeat courses, it may be necessary to include TERM in order to get an adequate primary key.

In some cases, a set-significant item may correspond to a primary key item in the owner record that is itself set-significant. Figure 2-8 gives an example. An EMPLOYEE is identified by SS#, and a PROJECT by by the DEPT that sponsors it and the SS# of its leader. There is one EMP-PROJ-LINK record for each employee currently working for each project. The use of a link record suggests that a project may employ many people, and that a person may work on several projects concurrently. Thus, to identify an EMP-PROJ-LINK record, one must know the employee's SS# as well as the identity of the project. Accordingly, all three set-significant items in the EMP-PROJ-LINK record participate in its primary key. A CONTRACT record, however, is uniquely determined by its CONTRACT#, so there is no need to specify additional primary key items.

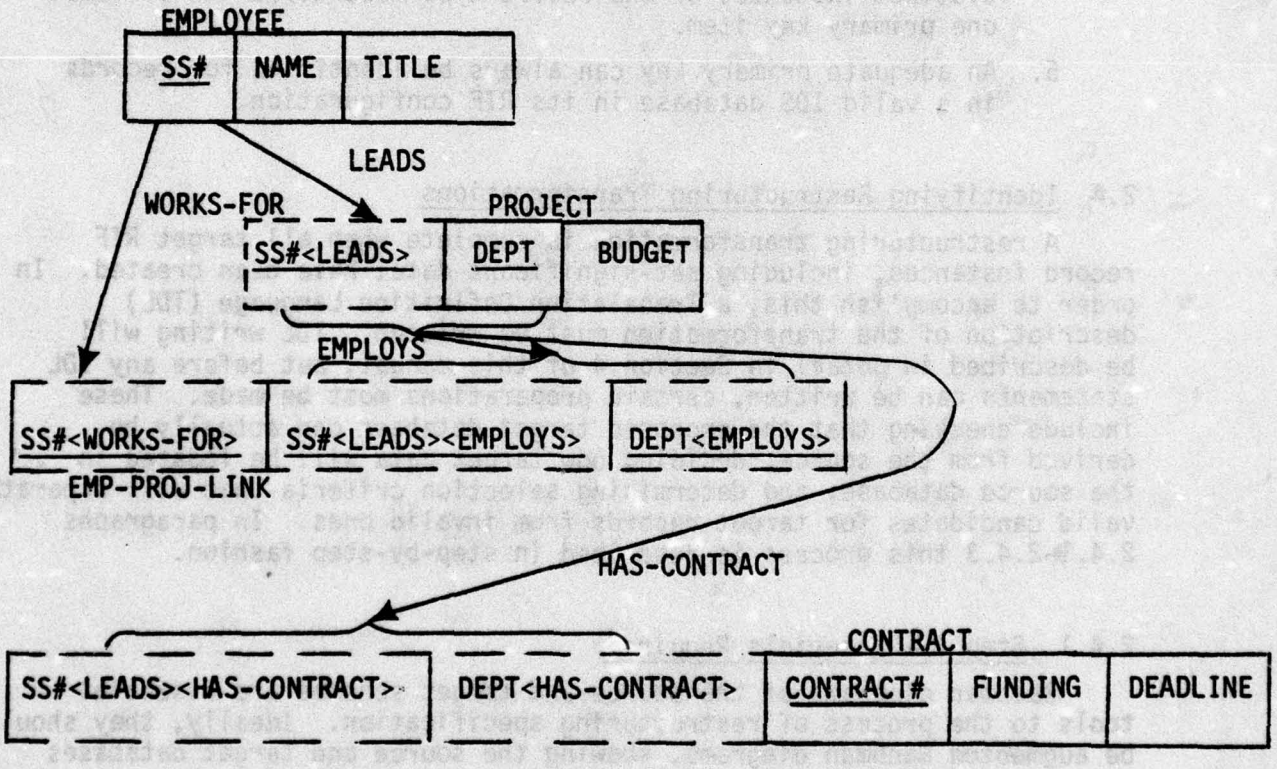


Figure 2-8
Set-significant Items that Correspond to
Set-significant Items

Set-significant primary key data is declared by a 61 level statement which identifies all sets whose set-significant items are to be primary key items in the member record type. Notice that this automatically enforces the following restriction; if one item significant to a particular set is also a primary key item, then all other items significant to that set are also primary key items.

Primary key identification may be summarized as follows:

1. For each record type, a collection of data items and sets must be declared as participants in the record's primary key. The declared data items and the items significant to the declared sets are referred to as the record's primary key items.
2. The record must be the valid member type for each set type.
3. None of the sets may be system-owned.
4. The combined values of the primary key items must uniquely determine an instance of the record type. That is, any two distinct instances of the record type must differ in at least one primary key item.
5. An adequate primary key can always be identified for records in a valid IDS database in its RIF configuration.

2.4 Identifying Restructuring Transformations

A restructuring transformation is complete when all target RIF record instances, including set-significant data, have been created. In order to accomplish this, a Translation Definition Language (TDL) description of the transformation must be written. TDL writing will be described in detail in Section 4 of this manual, but before any TDL statements can be written, certain preparations must be made. These include checking that the proposed target database can actually be derived from the source, deciding how target data will be located in the source database, and determining selection criteria that will separate valid candidates for target records from invalid ones. In paragraphs 2.4.1-2.4.3 this process is described in step-by-step fashion.

2.4.1 Step 1 - Materials Required

Bachman diagrams of the source and target schemas are essential tools to the process of restructuring specification. Ideally, they should be augmented Bachman diagrams, showing the source and target databases in their RIF configuration, complete with the set-significant items identified and named by the IDS Analyzer. However, it may not always be possible to complete the 61 level additions to the source and target MD sections and obtain a valid IDS Analyzer run before drawing up preliminary restructuring specifications. When this is the case, augmented diagrams should still be used with set-significant items named by the user. These will be replaced by actual names as soon as the IDS Analyzer can be successfully run.

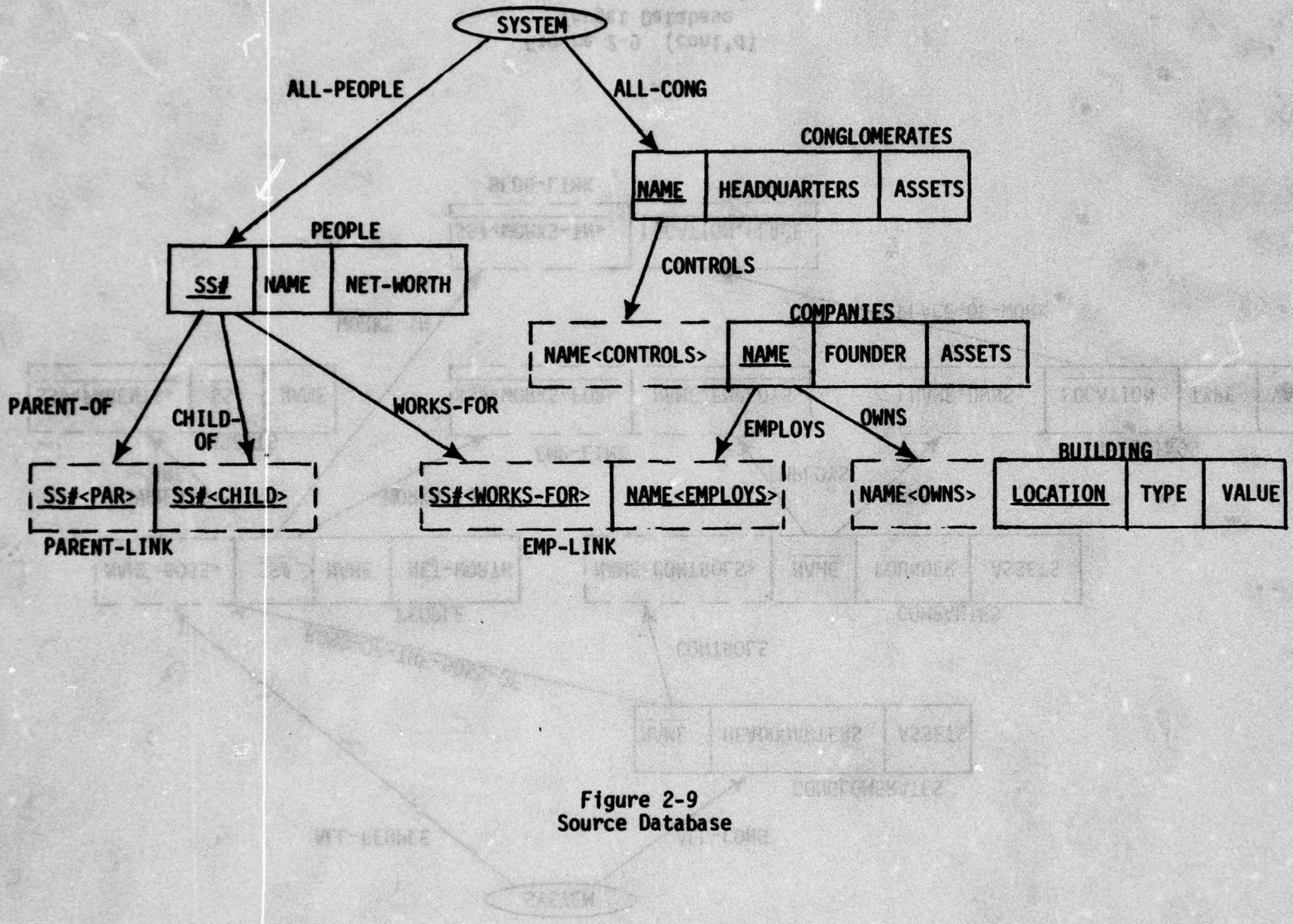


Figure 2-9
Source Database

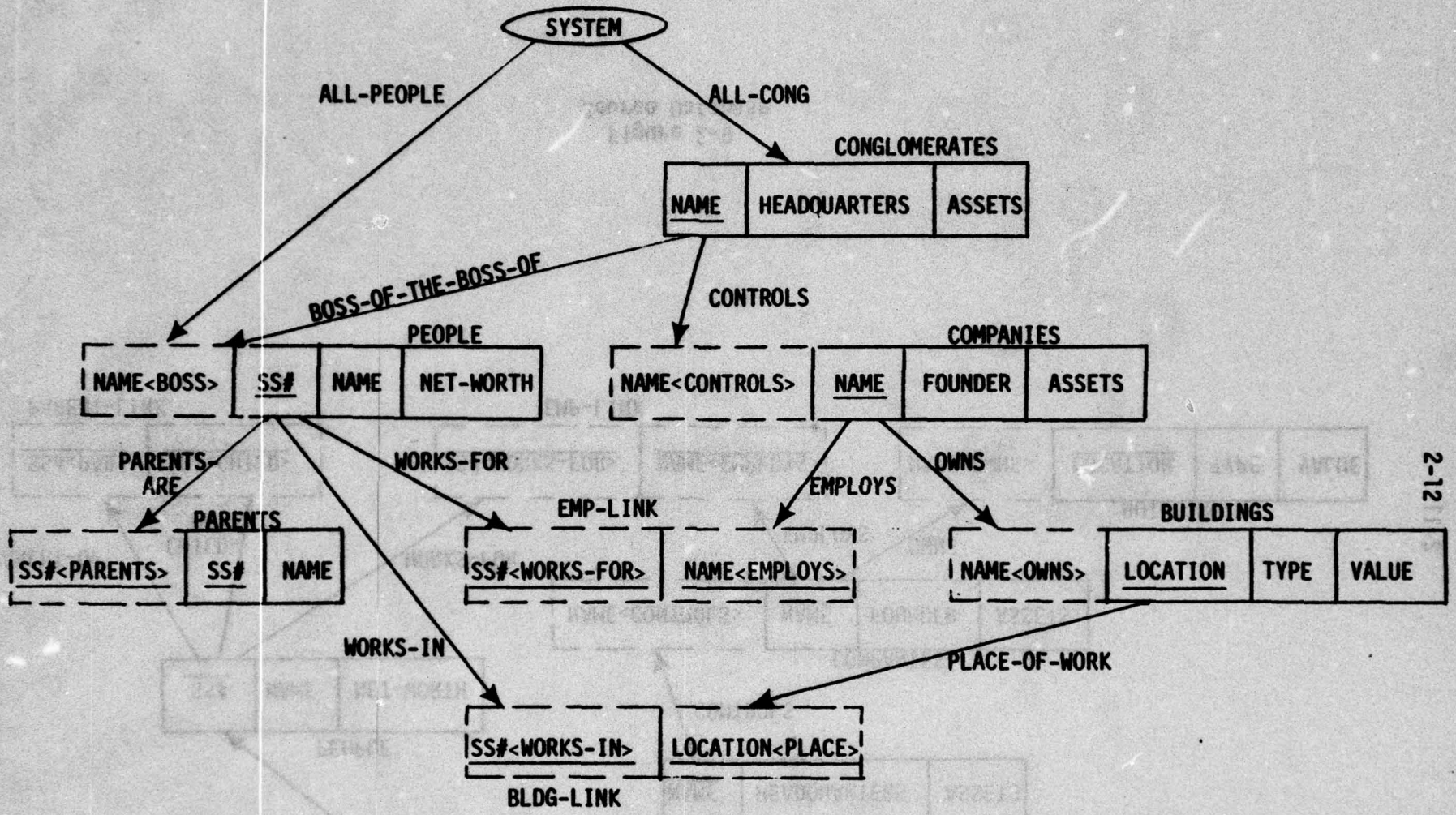


Figure 2-9 (cont'd)
Target Database

2.4.2 Step 2 - Semantic Validity of Proposed Restructuring

Restructuring cannot introduce new information into a database. Rather, during restructuring transformation, information is retrieved from its source representation and stored in its target representation. The information itself does not change. Thus, for a proposed restructuring to be semantically valid, every instance of every target record type must exist in some form in the source database. Furthermore, it must be distinguishable from other sets of source RIF instances of the same form which would not be valid instances of the proposed target record type. For example, consider the data structures shown in Figure 2-9. The target CONGLOMERATES, COMPANIES, EMP-LINK and BUILDINGS record types are certainly valid, since they already exist as source records. The target PEOPLE record is valid, provided that no person is permitted to work for two companies controlled by different conglomerates, in which case SS# would not be an adequate primary key. The form in which PEOPLE exists in the source is as follows; the tree in Figure 2-10 describes a subschema of the source database (non-directed arrows are used since the Restructurer traverses sets in both directions). Any instance of that tree defines a target PEOPLE record whose SS#, NAME, and NET-WORTH items are taken from the source PEOPLE record, and whose NAME<BOSS> item is the NAME item from the source CONGLOMERATES record.

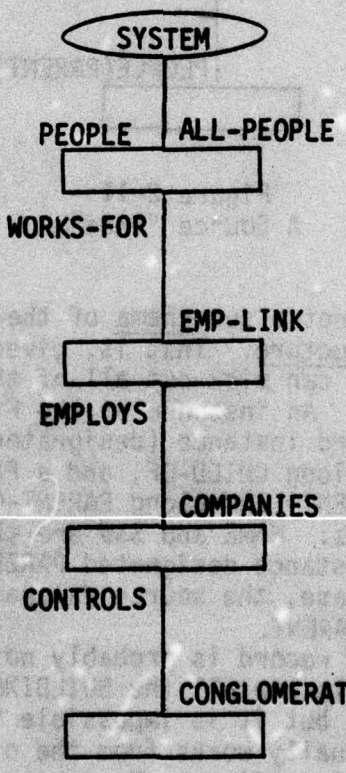


Figure 2-10
A Source Tree

It should be pointed out that the source RIF may contain multiple instances of some target PEOPLE records. This will occur whenever a person works for two or more companies.

The target PARENT record is also valid, but its representation in the source RIF is slightly less obvious than that of PEOPLE. Figure 2-11 shows what may be considered to be a tree in the source database.

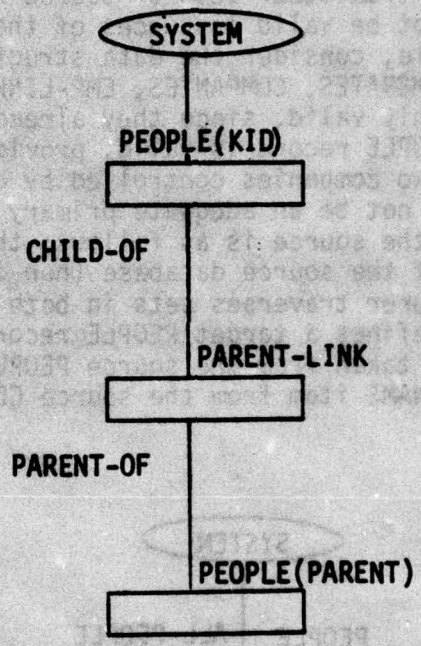


Figure 2-11
A Source "Tree"

Strictly speaking, it is not a subschema of the source database, but it could be called a sub-structure. That is, given a collection of instances of the source schema, one can pick out all of those that conform to the structure of Figure 2-11. An instance of the Figure 2-11 structure consists of a PEOPLE record instance (designated the "KID"), a PARENT-LINK record owned by the KID along CHILD-OF, and a PEOPLE record instance (the PARENT) that owns the PARENT-LINK along PARENT-OF. Such a trio represents a target PARENT as follows: NAME and SS# are the items of the same name from the source PEOPLE instance designated PARENT, and the SS#<PARENTS> is the KID's SS#. In this case, the source database contains exactly one instance of each target PARENT.

The target BLDG-LINK record is probably not valid, however. Given a PEOPLE record, one can locate all the BUILDINGS owned by companies for which the employee works, but it is impossible to distinguish buildings in which the employee actually works from the others owned by the same company. That is, the information to be represented by the BLDG-LINK record is not represented anywhere in the source database.

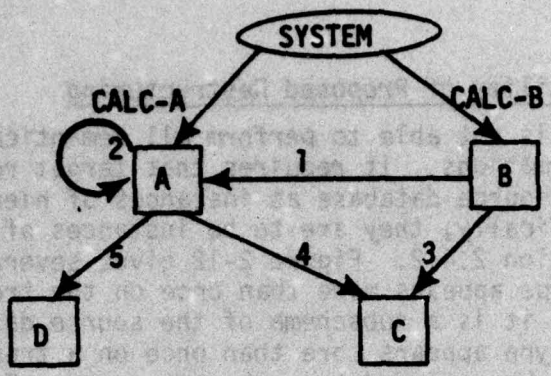
2.4.3 Step 3 - Feasibility of Proposed Restructuring

The Restructurer is not able to perform all semantically valid restructuring transformations. It requires that target record types be represented in the source database as instances of hierarchical substructures. Specifically, they are to be instances of trees of the type discussed in Section 2.4.2. Figure 2-12 gives several examples. If no source record type appears more than once on the tree (Figure 2-12 a, b, and c) then it is a subschema of the source database. Whenever a source record type appears more than once on a tree, then an identifying label must be assigned to each appearance. For instance, in Figure 2-12 d, the identifiers FIRST, SECOND, and THIRD have been attached to the three appearances of record type A on the tree. These identifiers are necessary. An instance of the tree 2-12d consists of an instance of A (the FIRST), the instance of B that owns it along set 1, the instance of A (the SECOND) that owns it along set 2, and the instance of A (the THIRD) that owns the SECOND along set 2. The three instances of A will, in general, be different, and references to them in restructuring specification will have to specify the instance of interest. The identifiers are often conceptual aids, as well, when they describe the role that a source record instance plays in the structure of a target record instance. This is the case with the KID and PARENT identifiers in Figures 2-11 and 2-16. Figures 2-12e and f give additional examples of the use of identifiers.

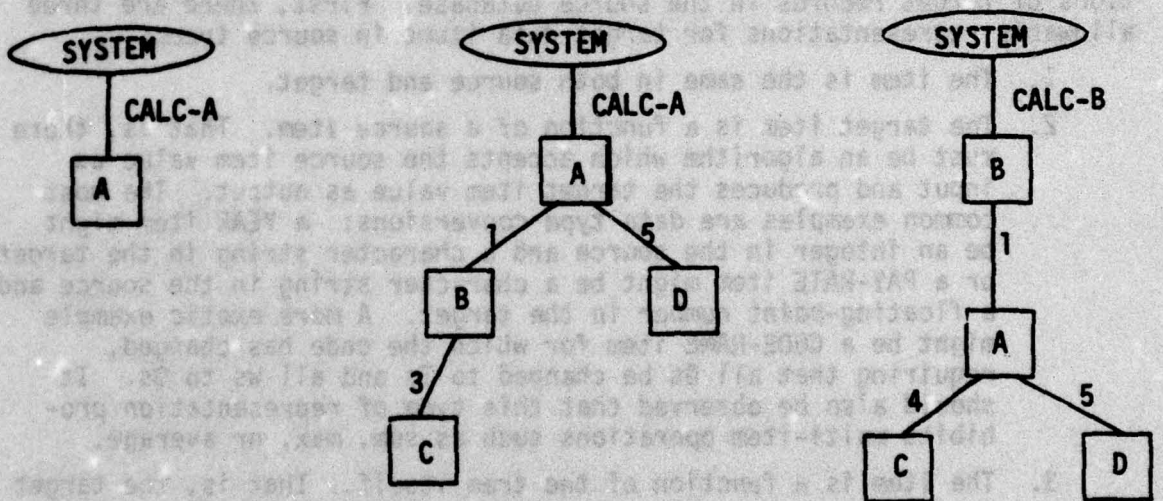
The Translator places two additional restrictions on the representations of target records in the source database. First, there are three allowable representations for target data items in source trees:

1. The item is the same in both source and target.
2. The target item is a function of a source item. That is, there must be an algorithm which accepts the source item value as input and produces the target item value as output. The most common examples are data type conversions: a YEAR item might be an integer in the source and a character string in the target, or a PAY-RATE item might be a character string in the source and a floating-point number in the target. A more exotic example might be a CODE-NAME item for which the code has changed, requiring that all Gs be changed to Ts and all Ws to Qs. It should also be observed that this type of representation prohibits multi-item operations such as sum, max, or average.
3. The item is a function of the tree itself. That is, the target item has the same value for every instance of the tree in the source RIF. An example of this is shown in Figure 2-15, which will be described shortly.

The second additional restriction requires that the criteria used to separate valid instances of a tree from invalid ones must involve only comparisons between an item on the tree and a constant value or one other item on the tree. Comparisons involving multi-item computations, such as average, are not permitted.

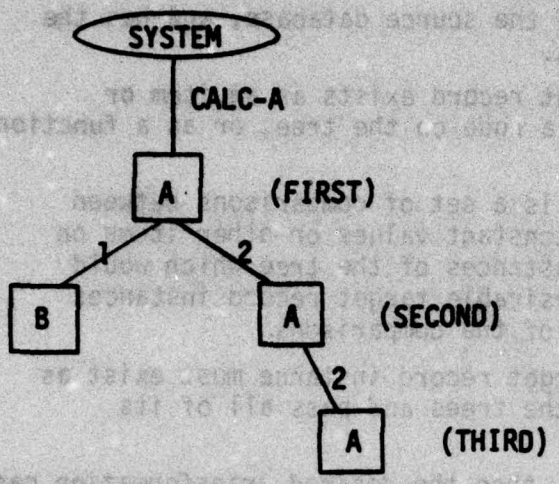


Source Structure

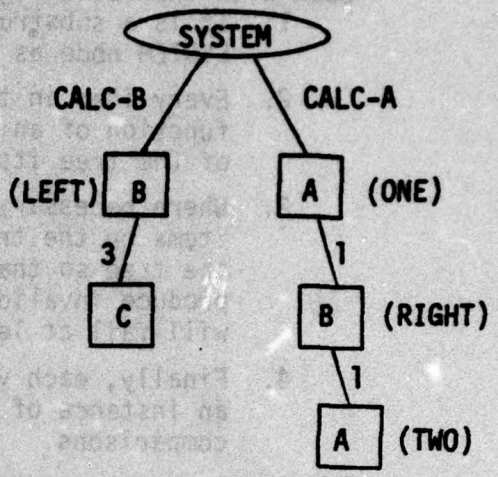


a. b. c.

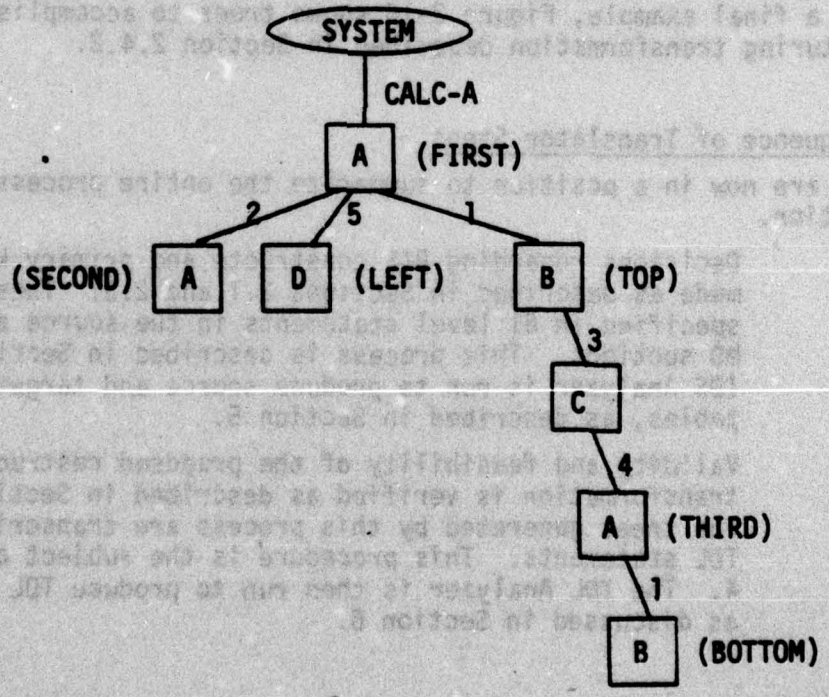
Figure 2-12
Schema and Hierarchical Substructures



d.



e.



f.

Figure 2-12 (cont'd)
Schema and Hierarchical Substructures

Step 3 may be summarized as follows:

For each target record type, construct a collection of trees, each with the following properties:

1. It is a substructure of the source database, and has the SYSTEM node as its root.
2. Every item in the target record exists as an item or function of an item in a node on the tree, or as a function of the tree itself.
3. Where necessary, there is a set of comparisons between items on the tree and constant values or other items on the tree so that all instances of the tree which would produce invalid or undesirable target record instances will fail at least one of the comparisons.
4. Finally, each valid target record instance must exist as an instance of one of the trees and pass all of its comparisons.

If step 3 can be accomplished, then the desired transformation can be performed by the Data Translator. If not, the transformation is beyond the Translator's capabilities.

For example, consider Figure 2-13. Suppose the database 2-13a is the source, and 2-13b is the target. Then trees, item correspondences, and qualification criteria to accomplish this transformation are shown in Figure 2-14. On the other hand, if we reverse the roles of source and target, then appropriate trees, item correspondences, and qualification criteria are shown in Figure 2-15. Notice that in each tree for the target KIDS record, the item SEX is a function of the tree itself, and not of any item on the tree.

As a final example, Figure 2-16 shows trees to accomplish the restructuring transformation described in Section 2.4.2.

2.5 Sequence of Translator Steps

We are now in a position to summarize the entire process of Data Translation.

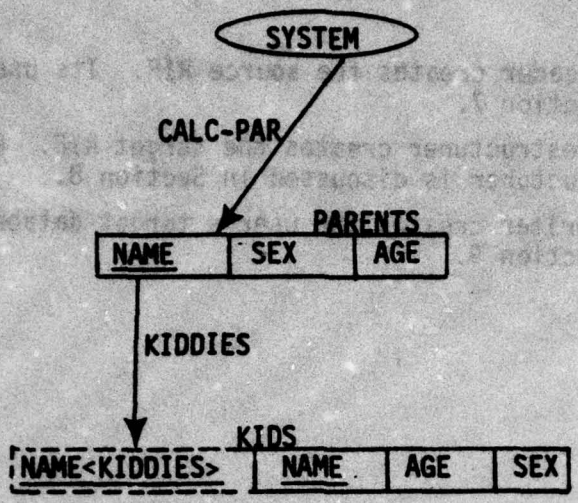
Step 1. Decisions regarding RIF constructs and primary keys are made as described in Sections 2.1 and 2.3. These are specified in 61 level statements in the source and target MD sections. This process is described in Section 3. The IDS Analyzer is run to produce source and target SDDL tables, as described in Section 5.

Step 2. Validity and feasibility of the proposed restructuring transformation is verified as described in Section 2.4. The trees generated by this process are transcribed into TDL statements. This procedure is the subject of Section 4. The TDL Analyzer is then run to produce TDL tables, as discussed in Section 6.

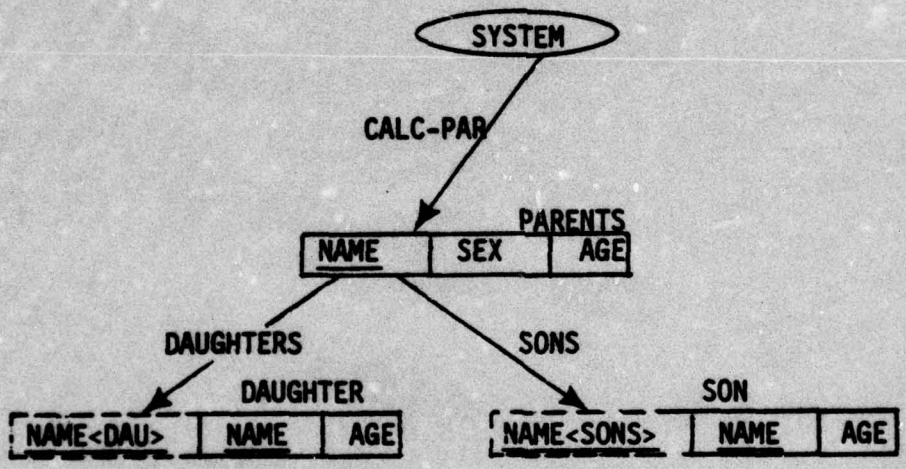
Step 3. The Reader creates the source RIF. Its use is described in Section 7.

Step 4. The Restructurer creates the target RIF. Running the Restructurer is discussed in Section 8.

Step 5. The Writer creates the user's target database, as described in Section 9.



a. A Database



b. A Related Database

Figure 2-13

Target Record	Tree	Items		Comparisons
	SYSTEM	Source	Target	none
PARENTS	CALC-PAR	NAME	→ NAME	
	PARENTS	SEX	→ SEX	
		AGE	→ AGE	
DAUGHTER	SYSTEM			
	CALC-PAR			
	PARENTS	NAME	→ NAME<DAU>	
	KIDDIES			
	KIDS	NAME	→ NAME	SEX="FEMALE"
		AGE	→ AGE	
SON	SYSTEM			
	CALC-PAR			
	PARENTS	NAME	→ NAME<SON>	
	KIDDIES			
	KIDS	NAME	→ NAME	SEX="MALE
		AGE	→ AGE	

Figure 2-14
A to B Restructuring

Target Record	Tree	Items (Source → Target)	Comparisons
PARENTS	SYSTEM → CALC-PAR → PARENTS → [ ]	NAME → NAME	
		SEX → SEX	none
		AGE → AGE	
KIDDIES	SYSTEM → CALC-PAR → PARENTS → [ ] → DAUGHTERS → DAUGHTER → [ ]	"FEMALE" → SEX	
		NAME → NAME<KIDDIES>	
			none
		NAME → NAME	
		AGE → AGE	
	SYSTEM → CALC-PAR → PARENTS → [ ] → SONS → SON → [ ]	"MALE" → SEX	
		NAME → NAME<KIDDIES>	none
		NAME → NAME	
		AGE → AGE	

Figure 2-15
B to A Restructuring

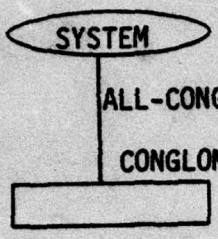
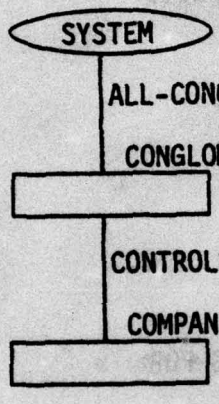
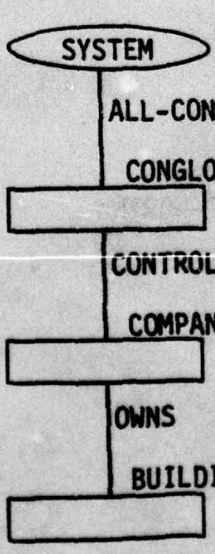
Target Record	Tree	Items (Source → Target)	Comparisons
CONGLOMERATES		NAME → NAME HEADQUARTERS → HEADQUARTERS ASSETS → ASSETS	none
COMPANIES		NAME → NAME<CONTROLS> NAME → NAME FOUNDER → FOUNDER ASSETS → ASSETS	none
BUILDINGS		NAME → NAME<OWNS> LOCATION → LOCATION TYPE → TYPE VALUE → VALUE	none

Figure 2-16

Trees, Items, Comparisons for the Transformation of Figure 2-1

Target Record	Tree	Items		Comparisons
		Source	Target	
EMP-LINK	SYSTEM			
	ALL-CONG			
	CONGLOMERATES			
	CONTROLS			
	COMPANIES	NAME	NAME<EMPLOYS>	none
	EMPLOYS			
	EMP-LINK			
	WORKS-FOR			
	PEOPLE	SS#	SS#<WORKS-FOR>	
PEOPLE	SYSTEM			
	ALL-PEOPLE			
	PEOPLE	SS#	SS#	
		NAME	NAME	
	WORKS-FOR	NET-WORTH	NET-WORTH	
	EMP-LINK			
	EMPLOYS			
	COMPANIES			none
	CONTROLS			
	CONGLOMERATES	NAME	NAME<BOSS>	

Figure 2-16 (cont'd)

Target Record	Tree	Items		Comparisons
		Source	Target	
PARENTS	SYSTEM			
	ALL-PEOPLE			
	PEOPLE(KID)	SS# →	SS#<PARENTS>	none
	CHILD-OF			
	PARENT-LINK			
	PARENT-OF			
	PEOPLE(PARENT)	SS# →	SS#	
		NAME →	NAME	

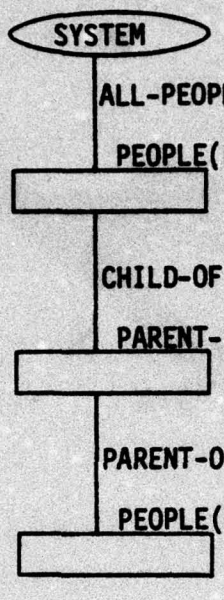


Figure 2-16 (cont'd)

3.0 AUGMENTING THE IDS MD WITH LEVEL 61 INFORMATION

In order to perform data translation between a source database and a target database, it is necessary to describe to the Translator the characteristics of the databases. Each of the Translator modules requires specific information about the data upon which it is executing. As designed, the Translator is table-driven in its execution. That is, from a description of the source and target databases, and guides for transforming source into target, the Translator can restructure almost any sequential, ISP or IDS database into a new IDS database. What the user must provide to the Translator is the aforementioned descriptions. This section outlines the rules for describing the source and target database.

The basis for the database description is an IDS MD section. This is true even if the source database(s) to be described is of the sequential or ISP type. For source IDS databases, the IDS MD already exists, whereas for source sequential and ISP databases, as well as the target IDS database, an IDS MD must be prepared. The user is referred to the IDS Programmer's Guide for aid in preparation of the MD section.

Once the MD sections have been written, it is necessary to extend them to explicitly provide information to the Translator that is otherwise unobtainable. Specifically, the IDS MD does not:

1. Identify a record's primary keys which must be known by the Restructurer.
2. Identify where in the database schema entry can be made (location of the tops of the database).
3. Identify unusual relations implemented without the use of IDS chains. For example, phantom pointers must be explicitly declared.
4. Identify ISP and sequential record identifiers that are required under the WWDMS B-3, T-1 and T-2 implementations.

In addition to the above, the user must restate some information that appears already to be in the IDS MD. This restatement is required because of the algorithm used by the Translator to get information from the MD to its internal data description tables (SDDL tables). Section 3.1 details the process of extending the IDS MD.

3.1 Process of Extending the IDS MD Section

To understand the rationale for extending the IDS MD it is necessary to be aware of the implementation of the IDS Analyzer. Briefly, the IDS MD section with extensions is passed to the IDS Translator (the Honeywell compiler) which is run in query mode. The resultant output is the IDS Query dictionary whose format is known. Using the IDS Query dictionary as input, the IDS Analyzer produces SDDL tables. The user is referred to the IDS Data Query Installation Guide for a complete description of the layout of the Query dictionary. It can be seen from the diagram of the dictionary schema in that document that information above and beyond the normal 01, 02 and 98 level entries of an MD section can be stored within the Validation and Description records. Figure 3-1 is an overview of the MD extension process.

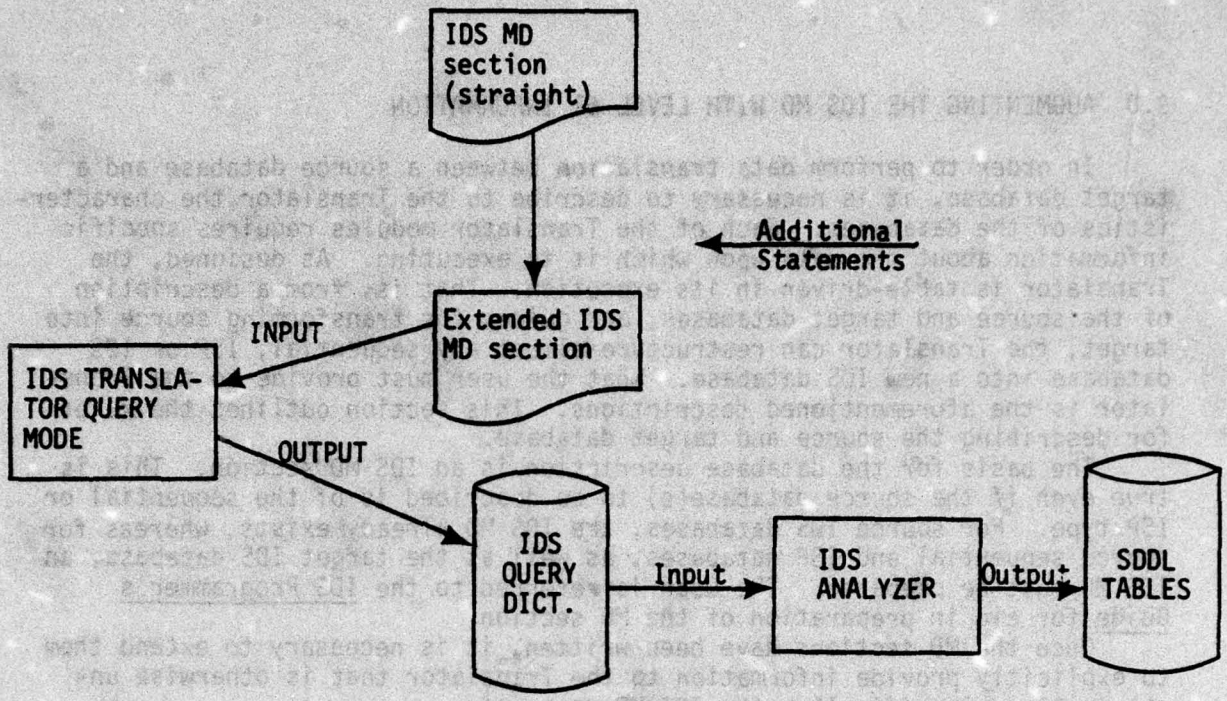


Figure 3-1
Extension of an IDS MD

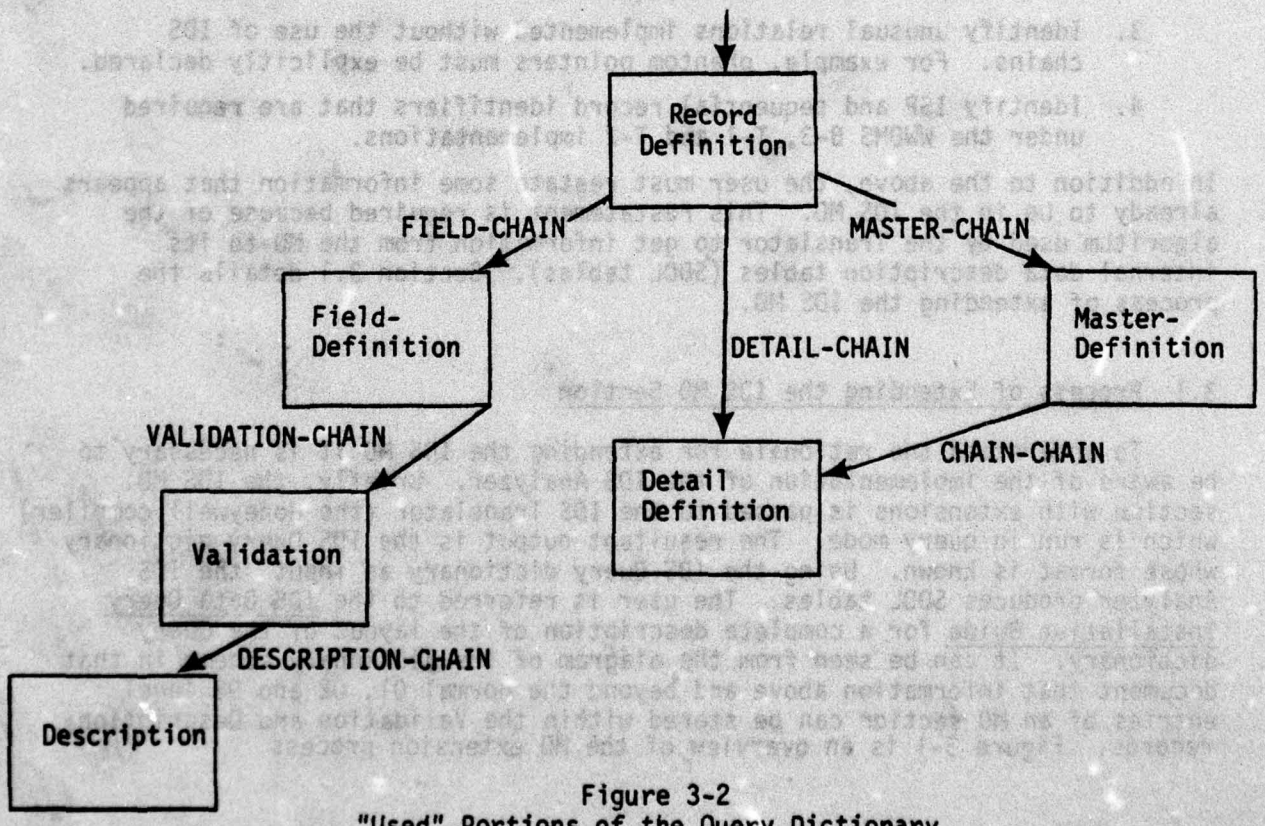


Figure 3-2
"Used" Portions of the Query Dictionary

Refer to Figure 3-2 to further examine the algorithm of conversion between MD section and SDDL tables. The diagram is a subset of the entire Query dictionary schema. Only the actual records used by the IDS Analyzer are indicated. The table below shows how information required in the SDDL tables is derived from the Query dictionary.

SDDL table information	Derivable from which Query dictionary records
1. Names and attributes of 01 records	1. Record definition
2. Names and attributes of non-01 records, e.g., contained-in-repeating groups	2. Validation and description
3. Names and attributes of 02 fields	3. Field definition
4. Names and attributes of 03-49 fields	4. Validation and description
5. Primary key information	5. Description
6. Match-key and phantom pointer relations	6. Description
7. Record identifiers for sequential and ISP files	7. Description
8. Relations implemented through chains	8. Master definition and detail definition
9. ISP and Sequential file relations	9. Master definition and detail definition.

As can be seen from the above table, obtaining information from only the Record definition, Field definition, Master definition and Detail definition records is insufficient to provide all the data to the SDDL tables. Validation records are automatically created by the IDS Translator in query mode for all 02-49 entries. The user, therefore, need not be concerned with explicit creation of these records. However, to create a description record, the user must code a level 61 entry in the IDS MD. Level 61 entries form the basis of all extensions to the MD section.

When coding level 61 entries, they must be placed within the MD section in accordance with the rules defined by the IDS Translator in query mode and the IDS Analyzer. Complete syntactical rules are given in Section 3.2. In general, an extended MD section will look like Figure 3-3. Level 61 entries are coded beneath the 02-49 field or group entry to which they apply.

```
01  record entry
    02  field entry
    02  field entry
    02  group entry
        03  group entry
        61  extension
            04 field entry
            04  field entry
        61 extension
    02  field entry
    61  extension
    61  extension
    61  extension
    61  extension
98  chain entry
98  chain entry

01  record entry
    02  field entry
    61  extension
    02  field entry
        .
        .
        .
```

Figure 3-3
A Generic Extended IDS MD Section

3.1.1 Further Rationale for Level 61 Entries

As the user reads further into this section, it is probable that he will feel that restructuring operations are being specified on the source database by denoting contained-in-repeating groups, phantom pointers and match-key relations. In a sense, it is true but these specifications must be performed at database description time (i.e., writing level 61s), otherwise the Restructurer, when executed, will not be aware of the non-IDS constructs that the user wishes to preserve. Consider it from the following perspective. The Restructurer uses as input not the source database, but a database produced by the Reader in which

1. all contained-in-repeating groups in the source user database are now full-fledged groups in the source Restructurer database (Source RIF).
2. all match-key and phantom pointer relations in the source user database are represented as legitimate sets in the Source RIF.

If the information represented by contained-in-repeating groups, phantom pointers and match-key relations is to be used at all in the construction of the target database, it must be identified by writing level 61 entries.

3.1.2 Summary of Required Extensions

Level 61 entries must be prepared for the following elements of a database.

1. All groups must have their primary keys identified.
2. All contained-in-repeating groups must be denoted.
3. Any match-key or phantom relations must be identified if the user desires the information represented in those relations to be preserved.
4. ISP and Sequential databases must have an item identified within each record type as the item containing the record ID.
5. Items whose usage is DISPLAY-1 or DISPLAY-2 must be identified.
6. Repeating groups that are to be ignored as groups must be identified.

3.2 Level 61 Rules

Before coding any extensions to the IDS MD section, certain rules and restrictions placed both by the IDS Analyzer and IDS Translator in query mode must be rigidly obeyed. These rules can be broken down into three categories - global rules pertaining to the IDS MD section, global rules pertaining to level 61 entries, and rules concerning automatic generation of names by the IDS Analyzer. These rules are described below.

3.2.1 Global MD Section Rules

1. Before making any extensions, the IDS MD section prepared must be legal and error-free. It is a good idea to process the MD section by the IDS Translator (not in query mode) and by COBOL to determine if it is indeed syntactically and semantically correct. A dummy COBOL-IDS program with a PROCEDURE-DIVISION of only a STOP RUN is ideal for this purpose.

2. All non-elementary item entries within the MD section must have a SIZE clause on them. For example, the MD section on the left below must be changed to the MD section on the right.

```
.                                .
.                                .
.                                .
02  GROUP-NAME.                 02  GROUP-NAME SIZE 4.
    03  SUB-GROUP-NAME.             03  SUB-GROUP-NAME SIZE 4.
        04  ITEM-1  PIC XX.             04 ITEM-1 PIC XX.
        04  ITEM-2  PIC XX.             04 ITEM-2 PIC XX.
02 ITEM-NAME  PIC XX.            02 ITEM-NAME  PIC XX.
.                                .
.                                .
.                                .
```

To compute the size of a group in COBOL use the following rules.

a. Single-precision computational items require preceding slack characters so that they start on a fullword boundary.

b. Double-precision computational items require preceding slack characters so that they start on a doubleword boundary.

c. Non-computational items never require preceding slack characters unless they are synchronized.

d. If the first item of the group is a computational item, then any required preceding slack characters are not counted as part of the size of the group.

e. The number of slack characters at the end of group occurrence is (if the group has an OCCURS clause):

condition	number of slack characters at end of group
i) group has no computational items within it	zero
ii) group has at least one single-precision computational but no double-precision computational items	However many slack characters needed such that the size of one occurrence of the group is evenly divisible by 6

iii) group has at least one double-precision computational item — However many slack characters needed such that the size of one occurrence of the group is evenly divisible by 12.

f. All records are assumed to start on a doubleword boundary.

3. If an OCCURS and a SIZE clause are coded on the same entry, the OCCURS clause must come first.

RIGHT	WRONG
03 GROUP-NAME OCCURS 2 TIMES SIZE 18.	03 GROUP-NAME SIZE 18 OCCURS 2 TIMES.

4. If a PICTURE and an OCCURS clause are coded on the same entry, the PICTURE clause must precede the OCCURS clause.

RIGHT	WRONG
03 REPEATING-ITEM PIC X(5) OCCURS 3 TIMES.	03 REPEATING-ITEM OCCURS 3 TIMES PIC X(5).

Note that "5" is the length (size) of one occurrence of REPEATING-ITEM.

5. A PICTURE and a SIZE clause cannot be coded on the same entry.

WRONG

```
03  REPEATING-ITEM PIC X(6) OCCURS 2 TIMES SIZE 12.
```

6. It is highly advisable to ensure that within one MD section all item (level 03-49), record and chain names are unique. This is more restrictive than absolutely necessary, but if non-unique item names are present, the IDS Translator in query mode will append a letter to the end of the name to make it unique. This adds an unnecessary level of indirection between the user's names and those names used by the Data Translator.

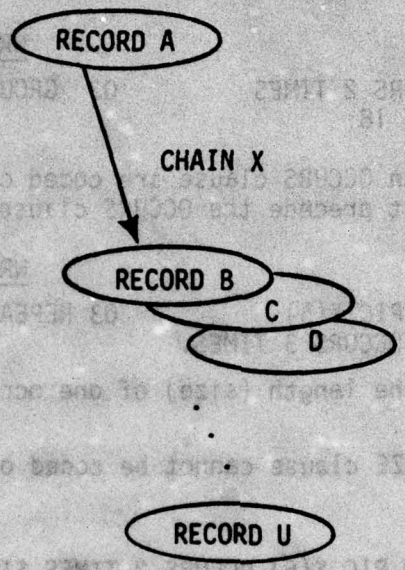
NOT RECOMMENDED	RECOMMENDED
02 FIRST-GROUP SIZE 10.	02 FIRST-GROUP SIZE 10.
03 ITEM-1 PIC XX.	03 ITEM-1-FG PIC XX.
03 ITEM-2 PIC X(8).	03 ITEM-2-FG PIC X(8).
02 SECOND-GROUP SIZE 5.	02 SECOND-GROUP SIZE 5.
03 ITEM-1 PIC XXX.	03 ITEM-1-SG PIC XXX.
03 ITEM-2 PIC XX.	03 ITEM-2-SG PIC XX.

7. Every 01 record must have as one of its 02 fields an entry exactly like: "02 TRANSLATION-INFORMATION SIZE 0", or abbreviated: "02 T-I SIZE 0". Beneath this 02 entry will be all primary key and relation information that must be added to the IDS MD section.

8. No group may have more than sixty items defined for it.

9. No item may be of a length greater than 255 characters.

10. No more than twenty records may be details on a given chain, e.g., if record A is the master of chain X as shown, only up to twenty record types may be a detail on that chain. Note that this rule applies to record types, not instances.



11. Ninety five (95) record types and five (5) databases are the maximum allowable in any one extended IDS MD section (see Section 3.8).

12. User names may not be TDL reserved words. This is due to the parsing restrictions of the TDL Analyzer. The reserved word list includes:

ACCEPT	ID	OWNER/MEMBER
ACCESS	IF	QUALIFIED
ACTUAL	IN	RECORD
ALL	IS	RESECT
ARE	LE	SELECT
AS	LITERALLY	SET
ASSIGN	LT	SIGNIFICANT
BY	MACRO	SOURCE
CONVERT	MEMBER/OWNER	TARGET
DATA	NAME	TDLAP
EOF	NAMES	TO
EQ	NE	VALUE
FROM	NULL	VIA
GE	ORDER	WHEN
GT	OTHER	WITH

3.2.2 Global Level 61 Rules

1. A level 61 entry must be coded beneath 02-49 entries only. In general they are coded directly beneath the item or group that they are intended to extend.
2. A level 61 entry is of the form "61 text" where the "61" may start no further left than column 12; the text cannot extend past column 72.
3. There must be at least one intervening blank between "61" and the text.
4. The maximum length of any 61 level text is 24 characters. Blanks preceding the beginning of the text are not counted in the 24 character limit but trailing blanks are. Essentially, this makes every 61 level entry contain exactly 24 characters. If the user needs to code a name that is greater than 24 characters, two 61 level entries are required, such as:

```
                                        24th character
                                        ▼
61  ITEM-NAME-THAT-IS-EXTREM
61  ELY
```

5. The maximum length of any user name, group, item or relation is 30 characters.
6. A comma "," cannot be the first character of a level 61 entry.

   NOT ALLOWED

   61 , text

7. When processing the description record image of a level 61 entry, spaces (blanks) are ignored by the IDS Analyzer. Of course user names cannot have intervening spaces within them. This means that if a user name is to be split over two level 61 lines, the first line must contain exactly 24 characters of text.
8. The legal character set that can be coded within a level 61 entry is any COBOL character with the exception of "+".

3.2.3 IDS Analyzer Automatically-Generated Names

For a variety of implementation reasons, some of which are too complex for this document, item or chain names as specified within the extended IDS MD section are altered by the IDS Analyzer. Similarly, it is necessary to create names for items and relations so that certain information (which is not supplied by the user...a convenience feature) is identified. As an example of the former case, chains which have multiple detail record types (for example: chain-"HAS-CHILDREN" with "SONS" and "DAUGHTERS" as the detail record types) are transformed into multiple chains, each with one detail record type with the new chain names being slight extensions to the original (see rule #1). In the second case above, the relation between a group and a contained-in-repeating group is never explicitly declared by the user but is instead generated by the IDS Analyzer (see rule #3).

In general, the user must be aware of how the IDS Analyzer will alter, adjust, or create names. If the user wishes to refer to a name within a level 61 entry, the actual character-string to use must be the IDS Analyzer altered or created name. The examples following this section will make this clear.

1. IDS allows multiple detail record types per chain type. This implies that one relation name is used from the master record to all of its detail record types on a particular chain type. However, the Data Translator requires that each (master record, detail record) tuple that is related have a unique relation name. This necessitates slight modification to the chain name according to the following rule:

   a) The chain name from the master record to its detail record types is appended with the characters "/char" where "char" is a number 1-9, or a letter A-K.

   b) If the chain name is greater than 28 characters only the first 28 characters are used.

   c) The assignment of the tack on "/char" value is based on reading the IDS MD section top to bottom, e.g., the first 98 DETAIL entry for that chain will be "/1", the second will be "/2", etc.

Example

```
01  REC-1 ...
    .
    .
    .
98  SUPER-CHAIN CHAIN MASTER CHAIN-ORDER IS AFTER.
01  REC-2 ...
    .
    .
    .
98  SUPER-CHAIN CHAIN DETAIL SELECT CURRENT MASTER.
01  REC-3 ...
    .
    .
    .
98  SUPER-CHAIN CHAIN DETAIL SELECT CURRENT MASTER.
```

Will generate the following two relation names:

RELATION	MASTER	DETAIL
SUPER-CHAIN/1	REC-1	REC-2
SUPER-CHAIN/2	REC-1	REC-3

2. If a repeating item is to be expanded out of its containing record, then the resultant group name is the name of the original item, and the sole item of the new group has as its name one of the following:

a) The original item name appended with "/IT".

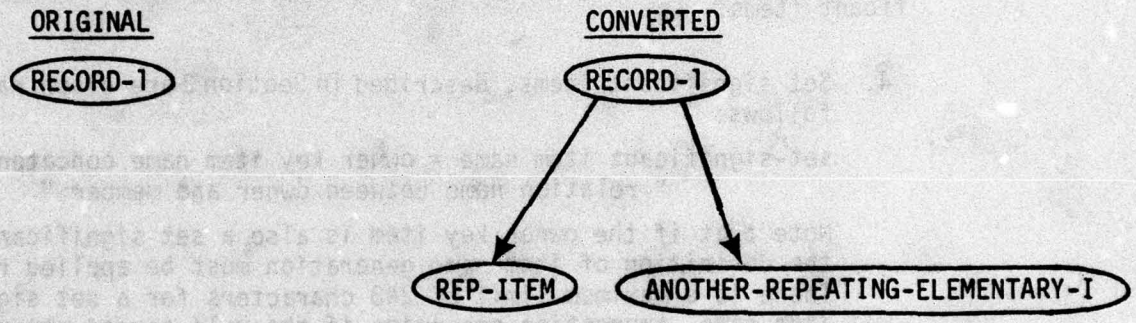
b) The first 27 characters of the original item name appended with "/IT" if the original name was longer than 27 characters.

Example

Suppose the MD section is as follows:

```
01  RECORD-1 TYPE IS 1 RETRIEVAL VIA ...
    02  GROUP-NAME SIZE 40.
        03  REP-ITEM  PIC X(5) OCCURS 4 TIMES.
        03  ANOTHER-REPEATING-ELEMENTARY-I  PIC X (10) OCCURS 2 TIMES.
    02  ...
```

If both of the 03 entries are to be considered as contained-in-repeating groups, then the IDS Analyzer will convert the schema represented by the above as follows:



The groups each will have a single item named as follows:

GROUP	ITEM NAME	LENGTH OF ITEM
REP-ITEM	REP-ITEM/IT	5
ANOTHER-REPEATING-ELEMENTARY-I	ANOTHER-REPEATING ELEMENTAR/IT	10

Note that RECORD-1 will no longer contain within it the 40 characters of GROUP-NAME. The relation between RECORD-1 and its new dependent groups is given a name according to rule 3 below.

3. Since all relations must have names, the relations between a group (record) and its contained-in-repeating groups (termed a concatenated relation) must be given names by the IDS Analyzer. The rule for construction of such names is as follows:

relation = the lesser of (the containing group and its first 12 characters) concatenated with "/OWNS/" concatenated with the lesser of (the contained-in-repeating group name and its first 12 characters)

Example

Using the names from the preceding example, the relation names that are assigned between RECORD-1 and its contained-in-repeating groups are:

RELATION	RELATION-NAME
Between RECORD-1 and REP-ITEM	RECORD-1/OWNS/REP-ITEM
Between RECORD-1 and ANOTHER-REPEATING-ELEMENTARY-I	RECORD-1/OWNS/ANOTHER-REPE

In Section 2 of this User Manual, the augmented Bachman diagram was introduced and explained. Since these diagrams are notable for their inclusion of set-significant items, it is necessary to identify and describe to the Data Translator their presence. However, all set-significant items are automatically created by the IDS Analyzer. To briefly review:

> A set-significant item for a member (detail) record type X is created for every primary key item in every owner (master) record type for which X is a member (detail). Note that set-significant items may be primary key items and hence, set-significant items can be created from set-significant items.

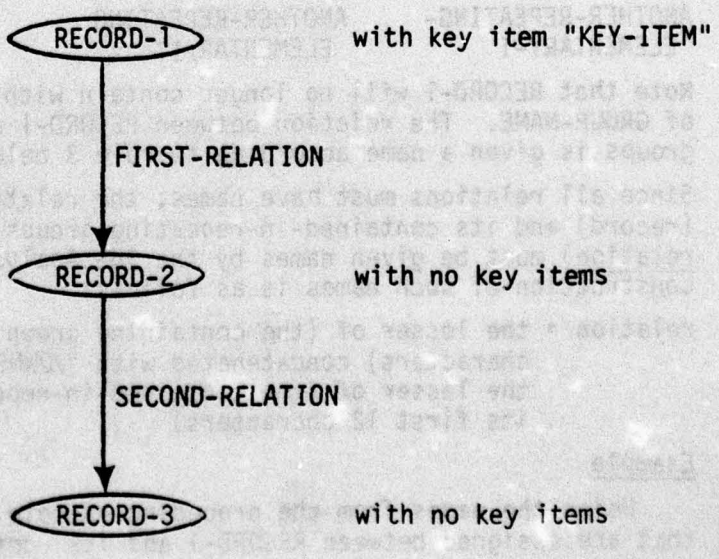
4. Set significant items, described in Section 2 are given names as follows:

   set-significant item name = owner key item name concatenated with "<relation name between owner and member>"

   Note that if the owner key item is also a set significant item, the definition of item name generation must be applied recursively. There is a maximum limit of 240 characters for a set significant item name, truncation occurring if the rule causes generation of length greater than 240.

Example

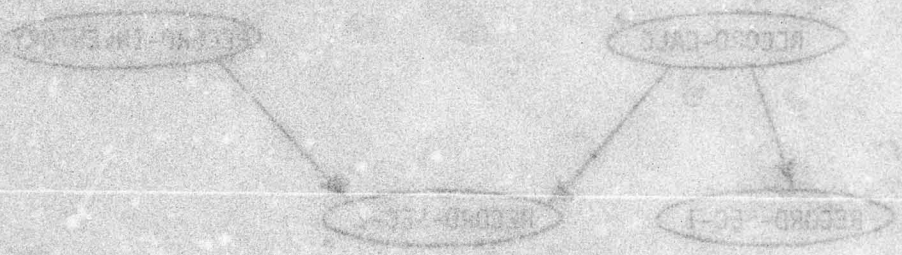
Suppose the following schema exists:



Primary keys for RECORD-2 must come from RECORD-1 and primary keys for RECORD-3 must come from RECORD-2. The keys for these records (which will be set-significant items) will have the following names.

RECORD	NAME OF SET-SIGNIFICANT ITEM (ALSO A KEY)
RECORD-2	KEY-ITEM<FIRST-RELATION>
RECORD-3	KEY-ITEM<FIRST-RELATION><SECOND-RELATION>

5. System entry points must be identified for the Data Translator. A system entry point is defined as that record which can be considered a detail of a record thought of as "SYSTEM". The relation between the "SYSTEM" and the system entry point record is named according to the following rules.



a) only CALC and PRIMARY records are defined to be system entry points.

b) the relation name is either

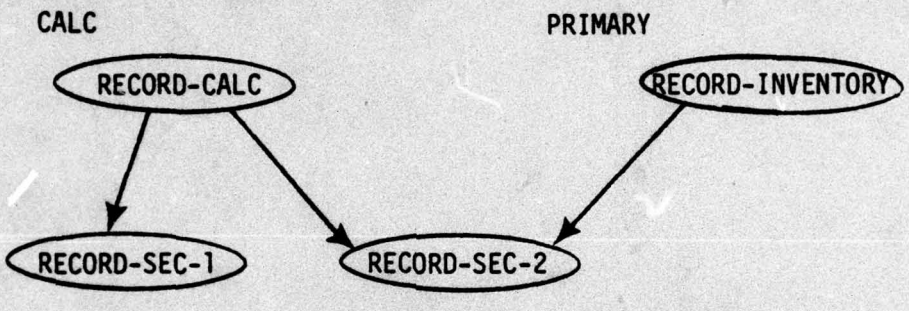
"CALC-" concatenated with the system entry point record name (up to 25 characters) if record is CALC

or

"PRIM-" concatenated with the system entry point record name (up to 25 characters) if record is PRIMARY

Example

Assume the following schema:



System entry point relations will be generated with names:

CALC-RECORD-CALC and PRIM-RECORD-INVENTORY

3.2.4 Complete Level 61 Syntax

This section details the correct syntax for all features of the level 61 entries. Specific rules and examples are given in Sectiond 3.3 through 3.6. Standard bracket, brace, double-bar and ellipsis notation is used. Key words are capitalized; generic user-names are in lower case. Because coding level 61 entries can be tedious, abbreviations are provided for certain key words. These abbreviations are denoted by underscored characters. For example, "PARENT" can be abbreviated as "PA". Strict attention should be paid to the use of the comma, colon, semi-colon and period within the syntax rules. Each separate level 61 entry (or entity) is denoted by a roman numeral.

Group definition

I. 61 OCCURS integer TIMES.

II. 61 DO-NOT-RESTRUCTURE.

III. 61 EOG.

Item definition

```
 I.  61  ITEM-INFORMATION.
II. [61  PAD-CHARACTER:  'character']
III.[61  DISPLAY: {1}                ]
                  {2}
```

Primary-key definition

```
 I.      02  TRANSLATION-INFORMATION SIZE 0.
         61  PRIMARY-KEYS.
         61  GROUP:  group-name-1,
       ||61  ITEMS:  item-name-1 [,item-name-2]...    .                          ||
       ||61  EXTERNAL-KEYS-FROM:  relation-name-1 [,relation-name-2]...    .     ||

     [   61  GROUP:  group-name-2,
       ||61  ITEMS:  item-name-1 [,item-name-2]...    .                          ||
       ||61  EXTERNAL-KEYS-FROM:  relation-name-1 [,relation-name-2]...    .     ||  ...
     ]
```

Relation-definition

```
 I.      02  TRANSLATION-INFORMATION SIZE 0.
         61  PHANTOM-POINTERS.
         61  RELATION:  relation-name-1,
         61  DEPENDENT:  group-name-1,
         61  POINTER:  item-name-1
        [61  ; RELATION:  relation-name-2,    ]
        [61    DEPENDENT:  group-name-2,      ]
        [61    POINTER:  item-name-2          ]  ...    .

II.     02  TRANSLATION-INFORMATION SIZE 0.
         61  MATCH-KEY-RELATIONS.
         61  PARENT:  group-name-1
         61  KEYS:  item-name-1 [,item-name-2]...    ;
         61  RELATION:  relation-name-1,
         61  DEPENDENT:  group-name-2,
         61  KEYS:  item-name-3 [,item-name-4]...
```

```
[ 61  ; PARENT:  group-name-3,                          ]
  61    KEYS:  item-name-5 [,item-name-6]...      ;
  61    RELATION:  relation-name-2,
  61    DEPENDENT:  group-name-4,
[ 61    KEYS:  item-name-7 [,item-name-8]...      ] ...   .
```

Sequential, ISP files

```
I.    61  ITEM-INFORMATION.
      61  IDENT:  'character-string'
```

3.3 Group definition

Within any 01 record, it is possible to have not only items, but also groups. Identification of these groups and an indication whether or not the group is to be considered a contained-in-repeating group must be given to the IDS Analyzer. Use of the following level 61 entries provides this information.

```
61  OCCURS integer TIMES.
61  EOG.
61  DO-NOT-RESTRUCTURE.
```

Rules:

1. Integer must be a positive integer.
2. Integer must be exactly equal to the COBOL OCCURS integer TIMES on the immediately preceding 02-49 entry.
3. Every OCCURS clause coded on a 02-49 entry must have a matching 61 OCCURS integer TIMES entry directly following.
4. Every 61 OCCURS integer TIMES entry must be matched by a 61 EOG (end-of-group) entry to be placed after the last item of the group.
5. If the group is not to be a contained-in-repeating group, then 61 DO-NOT-RESTRUCTURE must immediately follow the 61 OCCURS integer TIMES entry.
6. For groups whose number of occurrences is one, the COBOL OCCURS clause need not be coded. Three choices are available to the user.
   a) Code a 61 OCCURS 1 TIMES - 61 EOG. combination after the group entry and after the last item entry within the group. This will cause the group to become a contained-in-repeating group.
   b) Code a 61 DO-NOT-RESTRUCTURE after the group entry. This will cause the IDS Analyzer to treat the group as a single item made up of the concatenation of all of its subordinate entries.

c) Do nothing. The IDS Analyzer will treat all subordinate entries as items at the same hierarchical level as the group. The concept of a group is ignored.

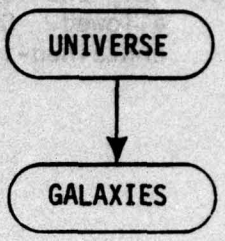
7. Contained-in-repeating groups may be defined for source databases only.

8. Only one level of nesting of contained-in-repeating groups is allowed. Contained-in-repeating groups cannot contain their own contained-in-repeating groups.

[Group definition]

Original Schema

Desired Translator-viewed Schema



Original MD Section

```
01 UNIVERSE TYPE IS 1 ...
   02 UNIVERSE-NAME  PIC X(10).
   02 CONTENTS SIZE 80.
      03 GALAXIES OCCURS 5 TIMES
                 SIZE 80.
         04 GALAXY-NAME  PIC X(10).
         04 GALAXY-SIZE  PIC 9(6)
   02 INHABITED-BY  PIC X(10).
```

Required Extended MD Section

```
01 UNIVERSE TYPE IS 1 ...
   02 UNIVERSE-NAME  PIC X(10).
   02 CONTENTS SIZE 80.
      03 GALAXIES OCCURS 5 TIMES
                 SIZE 80.
         61 OCCURS 5 TIMES.
            04 GALAXY-NAME  PIC X(10).
            04 GALAXY-SIZE  PIC 9(6).
         61 EOG.
   02 INHABITED-BY  PIC X(10).
```

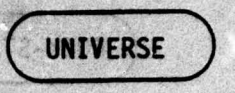
This example illustrates the simple expansion of one contained-in-repeating group from its contained-in group. Note that all groups have a SIZE clause on their entry. The 61 OCCURS 5 TIMES is placed immediately below the COBOL OCCURS clause while the 61 EOG. is placed after the last item of the group.

Because a contained-in-repeating group was defined, a relation name must be generated between UNIVERSE and GALAXIES. Rule 3 in Section 3.2.3 states that the relation name will be "UNIVERSE/OWNS/GALAXIES". As far as the Data Translator is concerned, UNIVERSE will not have any galaxy data within its record.

[Group definition]

Original Schema

Desired Translator-viewed Schema



Original MD Section

```
01 UNIVERSE TYPE IS 1 ...
   02 UNIVERSE-NAME  PIC X(10).
   02 CONTENTS SIZE 80.
      03 GALAXIES OCCURS 5 TIMES
                  SIZE 80.
         04 GALAXY-NAME  PIC X(10).
         04 GALAXY-SIZE  PIC 9(6).
   02 INHABITANTS  PIC X(10)
```

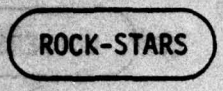
Required Extended MD Section

```
01 UNIVERSE TYPE IS 1 ...
   02 UNIVERSE-NAME  PIC X(10).
   02 CONTENTS SIZE 80.
      03 GALAXIES OCCURS 5 TIMES
                  SIZE 80.
      61 OCCURS 5 TIMES.
      61 DO-NOT-RESTRUCTURE.
         04 GALAXY-NAME  PIC X(10).
         04 GALAXY-SIZE  PIC 9(6).
      61 EOG.
   02 INHABITANTS  PIC X(10).
```

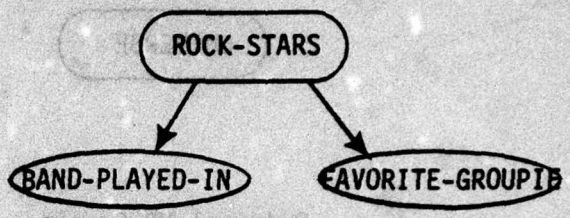
In this example, GALAXIES plays no part in the restructuring to the target database so a contained-in-repeating group is not desired. As before, the 61 OCCURS - 61 EOG. combination must be coded because a COBOL OCCURS clause is present. The 61 DO-NOT-RESTRUCTURE is placed immediately after the 61 OCCURS to indicate to the IDS Analyzer that GALAXIES is not a contained-in-repeating group but is to be treated as a single, elementary, item of alphanumeric type with a length of 80 characters.

[Group definition]

Original Schema



Desired Translator-viewed Schema



Original MD Section

```
01 ROCK-STARS TYPE IS 20 ...
   02 STARS-NAME  PIC X(20).
   02 BANDS SIZE 100.
      03 BANDS-PLAYED-IN  PIC X(20)
                          OCCURS 5 TIMES.
   02 FAVORITE-GROUPIE  SIZE 11.
      03 GROUPIE-NAME  PIC X(10).
      03 GROUPIE-SEX  PIC X.
```

Required Extended MD Section

```
01 ROCK-STARS TYPE IS 20 ...
   02 STARS-NAME  PIC X(20).
   02 BANDS SIZE 100.
      03 BANDS-PLAYED-IN  PIC X(20)
                          OCCURS 5 TIMES.
      61 OCCURS 5 TIMES.
      61 EOG.
   02 FAVORITE-GROUPIE  SIZE 11.
   61 OCCURS 1 TIMES.
      03 GROUPIE-NAME  PIC X(10).
      03 GROUPIE-SEX  PIC X.
   61 EOG.
```

This example illustrates two additional methods of identifying contained-in-repeating groups. BANDS-PLAYED-IN is a repeating item. FAVORITE-GROUPIE is a group that occurs only once. Note that for repeating items, the 61 OCCURS - 61 EOG. combination is placed directly beneath the repeating item entry. Upon completion of this coding, the IDS Analyzer will create the following:

GROUP	ITEMS
ROCK-STARS	STARS-NAME
BANDS-PLAYED-IN	BANDS-PLAYED-IN/IT (see rule 2 in Section 3.2.3)
FAVORITE-GROUPIE	GROUPIE-NAME, GROUPIE-SEX

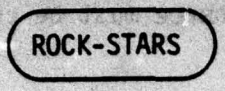
RELATIONS

ROCK-STARS/OWNS/BANDS-PLAYED

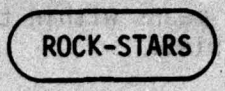
ROCK-STARS/OWNS/FAVORITE-GRO

[Group definition]

Original Schema



Desired Translator-viewed Schema



Original MD Section

```
01 ROCK-STARS TYPE IS 20 ...
   02 STARS-NAME  PIC X(20).
   02 BANDS SIZE 100.
      03 BANDS-PLAYED-IN  PIC X(20)
                          OCCURS 5 TIMES.
   02 FAVORITE-GROUPIE SIZE 11.
      03 GROUPIE-NAME  PIC X(10).
      03 GROUPIE-SEX  PIC X.
```

Required Extended MD Section

```
01 ROCK-STARS TYPE IS 20 ...
   02 STARS-NAME  PIC X(20).
   02 BANDS SIZE 100.
   61 DO-NOT-RESTRUCTURE.
      03 BANDS-PLAYED-IN  PIC X(20)
                          OCCURS 5 TIMES.
   61 OCCURS 5 TIMES.
   61 EOG.
   02 FAVORITE-GROUPIE SIZE 11.
      03 GROUPIE-NAME  PIC X(10).
      03 GROUPIE-SEX  PIC X.
```

This example illustrates the suppression of potential contained-in-repeating groups. Note however, that the 61 OCCURS - EOG. combination is required even though BANDS-PLAYED-IN will not become a group. Upon processing the extended MD section the IDS Analyzer will create the following:

GROUP	ITEMS
ROCK-STARS	STARS-NAME, BANDS, GROUPIE-NAME, GROUPIE-SEX

3.4 Item Definition

Two special item attributes must be explicitly described to the Data Translator via the IDS Analyzer. This information is unavailable through normal means within the Query dictionary and hence must be coded on level 61 entries.

```
61  ITEM-INFORMATION.
[61  PAD-CHARACTER:  'character']
[61  DISPLAY: {1}    ]
              {2}
```

Rules:

1. Character is any one character in the BCD set.
2. ITEM-INFORMATION followed optionally by PAD-CHARACTER and DISPLAY must be coded directly beneath the COBOL elementary item for which the special attributes apply.
3. PAD-CHARACTER: 'character' clause is used if the leading or trailing fill characters of the COBOL item are not the default blank for alphanumeric and zero for computational.
4. DISPLAY: $\left\{\begin{matrix}1\\2\end{matrix}\right\}$ is used if
   a) the COBOL elementary item has USAGE DISPLAY-1 in which case DISPLAY:1 is coded
   b) the COBOL elementary item has USAGE DISPLAY-2 in which case DISPLAY:2 is coded.

[Item definition]

Original Schema	Desired Translator-viewed Schema
INVENTORY	INVENTORY

Original MD Section	Required Extended MD Section
01 INVENTORY TYPE IS 1 ...	01 INVENTORY TYPE IS 1 ...
02 PART-NAME PIC X(10) USAGE DISPLAY-2.	02 PART-NAME PIC X(10) USAGE-DISPLAY-2.
02 PART-NUM PIC 9(8) USAGE DISPLAY-1.	61 ITEM-INFORMATION.
02 PART-PRICE PIC 9(6) COMP-1.	61 DISPLAY:2.
	02 PART-NUM PIC 9(8) USAGE DISPLAY-1.
	61 ITEM-INFORMATION.
	61 DIS:1.
Additional information	02 PART-PRICE PIC 9(6) COMP-1.
The characters before the first digit in PART-PRICE are always asterisks	61 ITEM-INFORMATION.
	61 PAD:'*'.

This example is a simple illustration of the placement of item definition level 61 entries. All occurrences of the 61 DISPLAY or 61 PAD-CHARACTER entries must follow the 61 ITEM-INFORMATION entry.

3.5 Primary Key Definition

The basis for the Restructurer algorithm is its requirement that every source and target group have primary keys within it such that each instance of a given group can be uniquely identified from all other instances. Primary keys for a group can come from either or both of two sources.

1. A group can wholly contain all item within it such that instances can be uniquely identified. CALC records are good examples since the fields used for randomization must have unique values for each record instance.
2. Only some of the items necessary to uniquely identify a group instance actually reside within the group. The remaining key items are contained in the owner group(s).

The following statements provide this information.

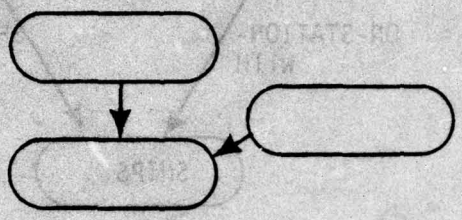
```
02  TRANSLATION-INFORMATION SIZE 0.
61  PRIMARY-KEYS.
61  GROUP:  group-name-1,
||61  ITEMS:  item-name-1  [,item-name-2] ...    .                    ||
||61  EXTERNAL-KEYS-FROM:  relation-name-1 [,relation-name-2] ...    . ||
[ 61  GROUP:  group-name-2,                                           ]
[||61  ITEMS:  item-name-1  [,item-name-2] ...    .                   ||]
[||61  EXTERNAL-KEYS-FROM:  relation-name-1 [,relation-name-2] ...   . ||] ...
```

Rules:

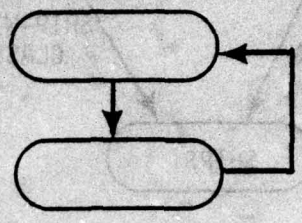
1. group-name-1, group-name-2.... must be groups defined within the current 01 record for which the 02 TRANSLATION-INFORMATION appears. Specifically, group-name-1 must be the 01 record name and group-name-2...group-name-n must be contained-in-repeating groups owned by group-name-1.
2. The item-name-1... beneath the 61 GROUP entry must be contained only within that group.
3. The relation-name-1, relation-name-2... immediately following 61 GROUP: group-name-n clause must be one of the following:
   a) the name of the chain for which group-name-n is a DETAIL (see rule 1 in Section 3.2.3 when the chain name is modified).
   b) the name of the concatenated relation generated where group-name-n is a member (see rule 3, Section 3.2.3).
   c) the name of a match-key relation for which group-name-n is a dependent.
   d) the name of a phantom pointer relation for which group-name-n is a dependent.

4. Loops or cycles of primary key derivation are not permitted. That is, if the arrows represent the paths where primary keys are defined via the EXTERNAL-KEYS-FROM clause, then the following illustrates legal and illegal cases of this rule.

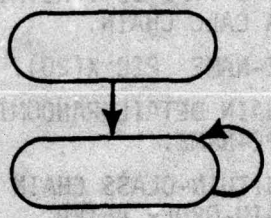
Legal



Illegal



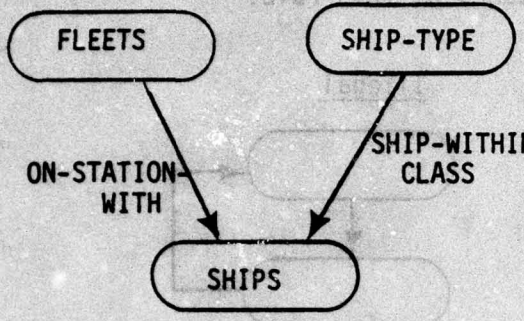
Illegal



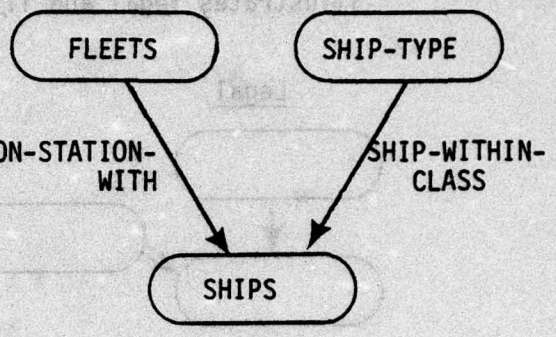
5. Every group defined by the user must have primary keys.
6. Groups at the "top" of a database, e.g., those groups that are not details of any non-CALC chain must have at least one key item and all of its key items must be present within that group.

[Primary Key definition]

Original Schema



Desired Translator-viewed Schema



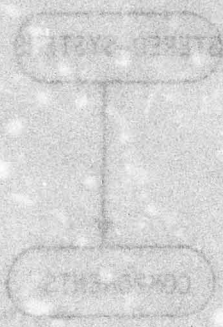
Original MD Section

```
01 SHIP-TYPE TYPE IS 1 RETRIEVAL
       VIA CALC CHAIN.
   02 TYPE-NAME  PIC X(20).
98 CALC CHAIN DETAIL RANDOMIZE ON
       TYPE-NAME.
98 SHIPS-WITHIN-CLASS CHAIN MASTER
       CHAIN-ORDER AFTER.

01 FLEETS TYPE IS 2 RETRIEVAL
       VIA CALC CHAIN.
   02 FLEET-DESG  PIC X(40).
98 CALC CHAIN DETAIL RANDOMIZE ON
       FLEET DESG.
98 ON-STATION-WITH CHAIN MASTER
       CHAIN-ORDER IS AFTER.

01 SHIPS TYPE IS 10 RETRIEVAL VIA
       SHIPS-WITHIN-CLASS CHAIN.
   02 SHIP-NAME  PIC X(15).
   02 SHIP-SKIPPER  PIC X(20).
98 ON-STATION-WITH CHAIN DETAIL
       SELECT CURRENT MASTER.
98 SHIPS-WITHIN-CLASS CHAIN DETAIL
       SELECT CURRENT MASTER.
```

Required Extended MD Section

```
01 SHIP-TYPE TYPE IS 1 RETRIEVAL
       VIA CALC CHAIN.
   02 TYPE-NAME  PIC X(20).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 PRIMARY-KEYS.
   61 GROUP:  SHIP-TYPE,
   61 ITEMS:  TYPE-NAME.
98 CALC-CHAIN DETAIL RANDOMIZE ON
       TYPE-NAME.
98 SHIPS-WITHIN-CLASS CHAIN MASTER
       CHAIN-ORDER IS AFTER.

01 FLEETS TYPE IS 2 RETRIEVAL
       VIA CALC CHAIN.
   02 FLEET-DESG  PIC X(40).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 P-K.
   61 G:  FLEETS,
   61 I:  FLEET-DESG.
98 CALC CHAIN DETAIL RANDOMIZE ON
       FLEET DESG.
98 ON-STATION-WITH CHAIN MASTER
       CHAIN-ORDER IS AFTER.
```

Original MD Section



Required extended MD Section

```
01 SHIPS TYPE IS 10 RETRIEVAL VIA
       SHIPS-WITHIN-CLASS CHAIN.
   02 SHIP-NAME  PIC X(15).
   02 SHIP-SKIPPER  PIC X(20).
   02 T-I SIZE 0.
   61 PRIMARY-KEYS.
   61 G:  SHIPS,
   61 I:  SHIP-NAME.
   61 E-K-F:
   61 SHIPS-WITHIN-CLASS.
98 ON-STATION-WITH CHAIN DETAIL
       SELECT CURRENT MASTER.
98 SHIPS-WITHIN-CLASS CHAIN DETAIL
       SELECT CURRENT MASTER.
```

Additional Information

The SHIP-NAME item with SHIPS is not sufficient to identify uniquely SHIPS instances. To do this requires knowing what type a given SHIP is.

This example shows the most straight-forward application of primary key definition. Since FLEETS and SHIP-TYPE are CALC records they are by definition uniquely identified by their randomize fields. SHIPS is a secondary record and requires the inclusion of the keys from one of its masters to uniquely identify a given SHIPS instance. In general, there is nothing within the MD section that states which items are primary keys; the user must know his/her database to define primary keys. Incorrect selection of primary keys will generally lead to incorrect target databases.

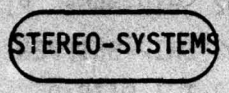
Note that the IDS Analyzer will generate two set significant items for the SHIPS record, one of which will be a primary key. These are (according to rule 4 in Section 3.2.3):

FLEET-DESG<ON-STATION-WITH>

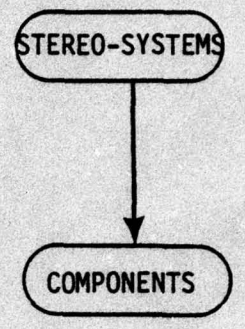
TYPE-NAME<SHIPS-WITHIN-CLASS> (primary key)

[Primary key definition]

Original Schema



Desired Translator-viewed Schema



Original MD section

```
01 STEREO-SYSTEMS TYPE IS 1 ...
   02 OWNER-NAME  PIC X(20).
   02 MADE-UP-OF SIZE 90.
      03 COMPONENTS OCCURS 5 TIMES
                   SIZE 90.
         04 COMP-NAME  PIC X(10).
         04 COMP-PRICE  PIC 9(6) COMP-1
   02 BOUGHT-FROM  PIC X(20).
      .
      .
      .
```

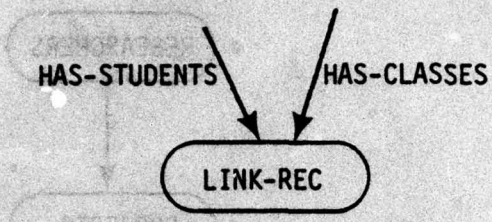
Required Extended MD Section

```
01 STEREO-SYSTEMS
   02 OWNER-NAME  PIC X(20).
   02 MADE-UP-OF SIZE 90.
      03 COMPONENTS OCCURS 5 TIMES
                   SIZE 90.
      61 OCCURS 5 TIMES
         04 COMP-NAME  PIC X(10).
         04 COMP-PRICE  PIC 9(6) COMP-1.
      61 EOG.
   02 BOUGHT-FROM  PIC X(20).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 PRIMARY-KEYS.
   61 G:  STEREO-SYSTEMS,
   61 I:  OWNER-NAME.
   61 G:  COMPONENTS,
   61 I:  COMP-NAME.
   61 E-K-F:
   61 STEREO-SYSTE/OWNS/COMPON
   61 ENTS.
```

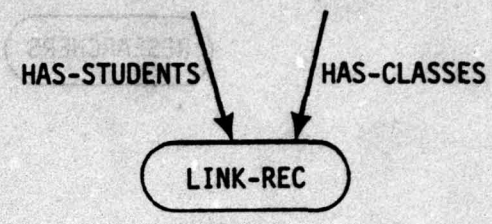
This example illustrates the basic principles for defining primary keys for contained-in-repeating groups. Note that the primary key of COMPONENTS is the combination of OWNER-NAME and COMP-NAME. The E-K-F clause uses the IDS Analyzer generated name for the concatenated realtion. Finally, observe that the size of COMPONENTS group is 90 which is not 5 x (10+6). Two slack characters are required at the end of each instance of components according to rule 2 of Section 3.2.1.

[Primary key definition]

Original Schema



Desired Translator-viewed Schema



Original MD Section

```
01 LINK-REC TYPE IS 3 RETRIEVAL VIA
       HAS-STUDENTS CHAIN.
98 HAS-STUDENTS CHAIN DETAIL
       SELECT CURRENT MASTER.
98 HAS-CLASSES CHAIN DETAIL
       SELECT CURRENT MASTER.
```

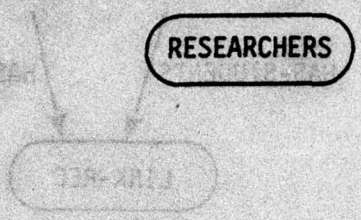
Required extended MD Section

```
01 LINK-REC TYPE IS 3 RETRIEVAL VIA
       HAS-STUDENTS CHAIN
   02 TRANSLATION-INFORMATION SIZE 0.
   61 G: LINK-REC,
   61 E-K-F:
   61 HAS-STUDENTS:
   61 HAS-CLASSES.
98 HAS-STUDENTS CHAIN DETAIL
       SELECT CURRENT MASTER.
98 HAS-CLASSES CHAIN DETAIL
       SELECT CURRENT MASTER.
```

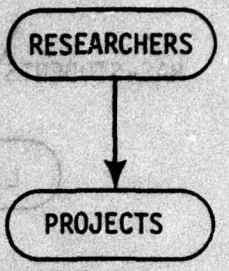
Many IDS databases contain relator or link records between two record types. A link record is required if a relationship exists both ways between the two record types. This example illustrates the declaration of primary keys for a link record. Note that every group (record) must have primary keys even if it does not contain any items. In this case, the primary keys of LINK-REC are the primary keys of LINK-REC's two master record types.

[Primary key definition]

Original Schema



Desired Translator-viewed Schema



Original MD Section

```
01 RESEARCHERS TYPE IS 1 RETRIEVAL VIA
       CALC CHAIN.
   02 RESEARCHER-NAME  PIC X(30).
   02 PROJECTS-WORKED-ON SIZE 50.
      03 PROJECTS  PIC X(10) OCCURS 5
                             TIMES.
98 CALC CHAIN DETAIL RANDOMIZE ON
       RESEARCHER-NAME.
```

Required Extended MD Section

```
01 RESEARCHERS TYPE IS 1 RETRIEVAL VIA
       CALC CHAIN.
   02 RESEARCHER-NAME  PIC(30).
   02 PROJECTS-WORKED-ON SIZE 50.
      03 PROJECTS  PIC X(10) OCCURS 5
                             TIMES.
      61 OCCURS 5 TIMES.
      61 EOG.
   02 TRANSLATION-INFORMATION SIZE 0.
   61 PRIMARY-KEYS.
   61 G:  RESEARCHERS,
   61 I:  RESEARCHER-NAME.
   61 G:  PROJECTS,
   61 I:  PROJECTS/IT.
   61 E-K-F:
   61 RESEARCHERS/OWNS/PROJECT
   61 S.
98 CALC CHAIN DETAIL RANDOMIZE ON
       RESEARCHER-NAME.
```

In this example, the user has chosen to make the repeating item "PROJECTS" a group for restructuring purposes. Because it is a group it must have primary keys declared for it. Assuming that more than one RESEARCHER could work on a given project, then to identify a PROJECTS record the PROJECTS/IT item and keys from RESEARCHERS (e.g., RESEARCHER-NAME) is required. Note that the item for the group PROJECTS is named according to the rules in Section 3.2.3 (rule #2).

3.6 Relation Definition

In normal IDS databases, relations between record types are implemented via the 98 CHAIN MASTER - CHAIN DETAIL combination. The IDS Analyzer will automatically store into the SDDL tables a relation for every CHAIN MASTER - CHAIN DETAIL pair with the relation name the same as the chain name (subject to rule 1 Section 3.2.3). However, because IDS is somewhat restrictive in terms of legal database schemas, certain users have implemented new relations outside the bounds or knowledge of the IDS software. For data translation purposes, two extraordinary constructs used to implement relations can be defined via level 61 entries. These constructs are match-key relations and phantom pointer relations.

3.6.1 Match-key Relations

Match-key relations exhibit the following characteristics:

1. There is no physical implementation of the relation via pointers. Instead, the relation exists between two record types if the values of items designated as the "match-keys" have identical values.
2. One record type of the relation is denoted to be the parent and the other record type is the dependent of the relation.
3. The parent and dependent record types may be the same record type. That is, for any one instance of that record type, its dependents are all other record instances of that type that possess the same match-key values as the parent.

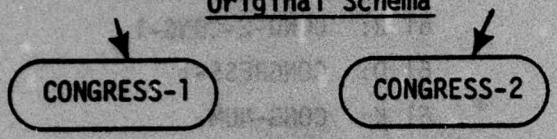
```
02 TRANSLATION-INFORMATION SIZE 0.
61 MATCH-KEY-RELATIONS.
61 PARENT:  group-name-1,
61 KEYS:  item-name-1 [,item-name-2] ...    ;
61 RELATION:  relation-name-1,
61 DEPENDENT:  group-name-2,
61 KEYS:  item-name-3 [,item-name-4] ...
[ 61; PARENT:  group-name-3,                  ]
[ 61 KEYS:  item-name-5 [,item-name-6] ...   ; ]
[ 61 RELATION:  relation-name-2,              ]
[ 61 DEPENDENT:  group-name-4,                ]
[ 61 KEYS:  item-name-7 [item-name-8] ...     ]  ...  .
```

Rules:

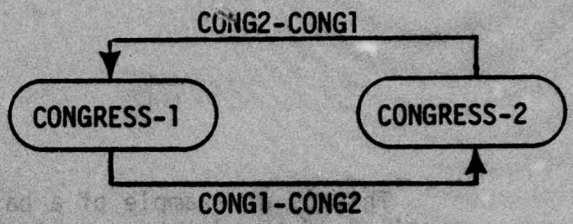
1. MATCH-KEY-RELATIONS must have as its immediately preceding 02 entry "02 TRANSLATION-INFORMATION SIZE 0". In general, MATCH-KEY-RELATIONS may either precede or follow the level 61 entries for primary keys and phantom pointers.
2. group-name-1, group-name-3, etc., must be a groups defined within the current 01 record where the MATCH-KEY-RELATIONS appears. This restricts the parent of a match-key relation to be either the 01 record name or a contained-in-repeating group defined within the current 01 record.
3. group-name-2, group-name-4, etc., may be any group within the entire level 61 extended MD section and is designated as the dependent of the relation.
4. relation-name-1, relation-name-2, etc., can be any name except for chain names or other names already used for concatenated, phantom pointer or match-key relations.
5. item-name-1, item-name-2, etc., must be items within the parent group.
6. item-name-3, item-name-4, etc., must be items within the dependent group of the relation.
7. The number and type of the items in the parent must be exactly the same as those of the dependent.
8. Multiple match-key relations can be defined within the same MATCH-KEY-RELATIONS section by placing a semicolon after the last key item in the dependent and then repeating the syntax as specified.
9. Match-key relations <u>cannot</u> be de defined for target databases.

[Match-key definition]

Original Schema



Desired Translator-viewed Schema



Original MD Section

```
01 CONGRESS-1 TYPE IS 1 ...
   02 CONG-NUM  PIC X(4).
   02 YEAR-START  PIC 9(4).
98 ...

01 CONGRESS-2 TYPE IS 2 ...
   02 CONGRESS-NUMB  PIC X(4).
98 ...
```

Required Extended MD Section

```
01 CONGRESS-1 TYPE IS 1 ...
   02 CONG-NUM  PIC X(4).
   02 YEAR-START  PIC 9(4).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 MATCH-KEY-RELATIONS.
   61 PARENT:  CONGRESS-1,
   61 KEYS:  CONG-NUM;
   61 RELATION:
   61   CONG-1-CONG-2.
   61 DEPENDENT:
   61   CONGRESS-2,
   61 KEYS:  CONGRESS-NUMB.
   61 PRIMARY-KEYS.
   61 G:  CONGRESS-1,
   61 I:  CONG-NUM.
98 ...

01 CONGRESS-2 TYPE IS 2 ...
   02 CONGRESS-NUMB  PIC X(4).
   02 T-I SIZE 0.
   61 P-K.
   61 G:  CONGRESS-2,
   61 I:  CONGRESS-NUMB.
   61 MATCH-KEY-RELATIONS.
```

Original MD Section

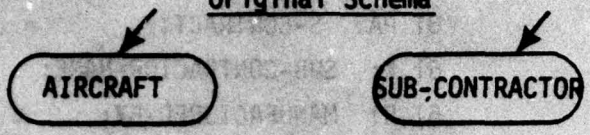
Required Extended MD Section

```
61 PA:  CONGRESS-2,
61 K:  CONGRESS-NUMB;
61 R:  CONG-2-CONG-1,
61 D:  CONGRESS-1
61 K:  CONG-NUM.
98 ...
```

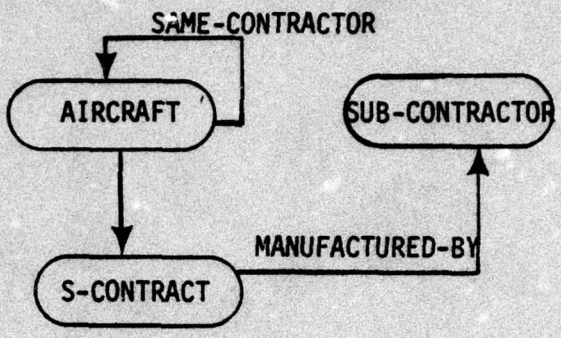
This is an example of a basic match-key relation application. Two sections of the database, unlinked by chains, are to be related through the value of the (CONG-NUM, CONGRESS-NUMB) tuple of key values. The result of this description to the Restructurer will be no different than if CONGRESS-1 and CONGRESS-2 were related via chains. Note that the level 61 entries for CONGRESS-2 utilize the legal MATCH-KEY-RELATIONS abbreviations.

[Match-key definition]

Original Schema



Desired Translator-viewed Schema

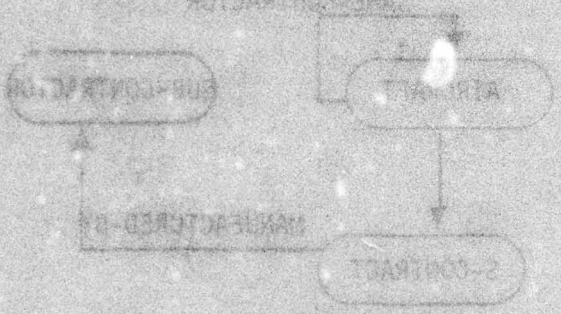


Original MD Section

```
01 AIRCRAFT TYPE IS 1 ...
   02 AIRCRAFT-NAME  PIC X(20).
   02 CONTRACTOR  PIC X(15).
   02 ALSO-MADE-BY SIZE 200.
      03 S-CONTRACT OCCURS 10 TIMES
                   SIZE 200.
         04 SUB-CONTRACTOR-NAME
                   PIC X(20).
98 ...

01 SUB-CONTRACTOR TYPE IS 2 ...
   02 SUB-NAME  PIC X(20).
   02 SUB-LOCALE  PIC X(40).
98 ...
```

Required Extended MD Section

```
01 AIRCRAFT TYPE IS 1 ...
   02 AIRCRAFT-NAME  PIC X(20).
   02 CONTRACTOR  PIC X(15).
   02 ALSO-MADE-BY SIZE 200.
      03 S-CONTRACT OCCURS 10 TIMES
                   SIZE 200.
      61 OCCURS 10 TIMES.
         04 SUB-CONTRACTOR-NAME
                   PIC X(20).
      61 EOG.
   02 T-I SIZE 0.
   61 P-K.
   61 G:  AIRCRAFT,
   61 I:  AIRCRAFT-NAME.
   61 G:  S-CONTRACT
   61 I:  SUB-CONTRACTOR-NAME.
   61 E-K-F:
   61 AIRCRAFT/OWNS/S-CONTRACT
   61 .
   61 MATCH-KEY-RELATIONS.
   61 PA:  AIRCRAFT,
   61 K:  CONTRACTOR;
   61 R:  SAME-CONTRACTOR,
   61 D:  AIRCRAFT
```



Original MD Section

Required Extended MD Section

```
61 K:  CONTRACTOR;
61 PA:  S-CONTRACT;
61 K:  SUB-CONTRACTOR-NAME;
61 R:  MANUFACTURED-BY;
61 D:  SUB-CONTRACTOR,
61 K:  SUB-NAME.
98 ...

01 SUB-CONTRACTOR TYPE IS 2 ...
   02 SUB-NAME  PIC X(20).
   02 SUB-LOCALE  PIC X(40).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 P-K.
   61 G:  SUB-CONTRACTOR,
   61 I:  SUB-NAME.
98 ...
```

This complex example illustrates the method for coding match-key relations under the following conditions:

1. Records AIRCRAFT and SUB-CONTRACTOR are not linked together by any chains or other relationships.
2. The user desires that all AIRCRAFT manufactured by the same manufacturer be linked together.
3. The user desires that the repeating information within AIRCRAFT on sub-contractors be removed from AIRCRAFT and placed into its own group. Furthermore, this group is to be related to the SUB-CONTRACTOR record via the sub-contractor's name.

Note in the example that several things must be accomplished via the level 61 entries.

a) Primary keys must, as always, be defined for all groups.

b) The contained-in-repeating group S-CONTRACT must be identified.

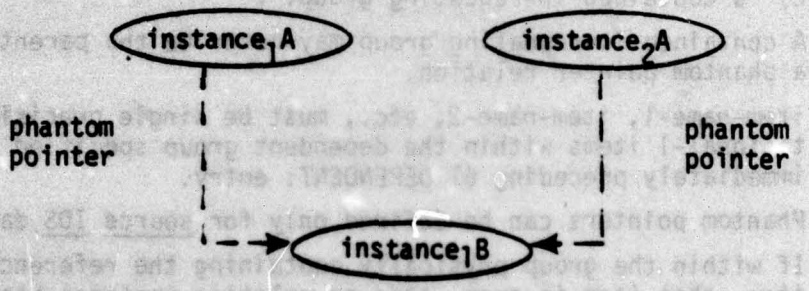
c) Match-key relations from both AIRCRAFT and S-CONTRACT are stated underneath the same 01 record (e.g., AIRCRAFT).

3.6.2 Phantom Pointer Relations

Of all extraordinary non-IDS constructs implemented by the user phantom pointer relations are the most difficult to describe via level 61 entries. A phantom pointer relation is restricted to the following definition:

1. If record or group A possesses a field whose value is that of a reference code to record or group B, then the relationship defined via that pointer is termed a phantom pointer relation.
2. A distinction is made between a phantom pointer and a phantom chain. The former is an implementation of a 1:1 relation between groups A and B while the latter is an implementation of a 1:n relation between group A and several instances of group B. Phantom Chains are not implemented as a feature of the Data Translator and hence cannot be described via level 61 entries.
3. For all purposes, if a pointer within group A points to an instance of group B, the relation between the two groups has group B as owner (or parent) and group A as member (or dependent). This is the exact reverse of how one would normally regard the relation.

   Suppose the following instance schema exists. Group A has a phantom pointer to group B.



   If a relation were declared to have A as owner and B as member, then the representation of this relation within the source RIF database would violate the fundamental law of network databases.

   "No member instance may be owned along the same relation type by more than one instance of a given owner group type. Because of this attribute of network databases, the relation must be viewed as though group B were the owner and group A the member."

4. Phantom pointers can be described only for IDS Source databases.

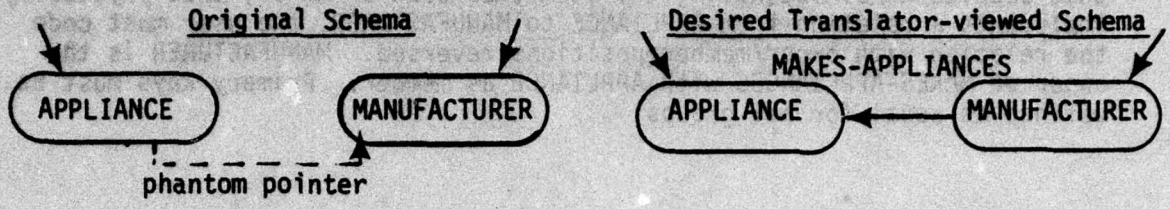
```
02 TRANSLATION-INFORMATION SIZE 0.
61 PHANTOM-POINTERS.
61 RELATION:  relation-name-1,
61 DEPENDENT:  group-name-1,
61 POINTER:  item-name-1
```

```
[ 61 ;RELATION:  relation-name-2,
  61  DEPENDENT:  group-name-2,
  61  POINTER:  item-name-2          ]  ...  .
```

Rules:

1. The PHANTOM-POINTERS level 61 entries must appear within the 02 TRANSLATION-INFORMATION for the group (record) that is considered to be the owner (parent) of the relation. See the #3 definition of a phantom pointer relation above.
2. relation-name-1, relation-name-2, etc., can be any previously unused chain name, concatenated relation name or match-key relation name.
3. group-name-1, group-name-2, etc., must be the name of the group in which the phantom pointer item physically resides. The dependent of a phantom pointer relation may be following:
   a) the same group type as the parent.
   b) a different 01 record than the parent.
   c) a contained-in-repeating group.
4. A contained-in-repeating group may never be the parent (owner) of a phantom pointer relation.
5. item-name-1, item-name-2, etc., must be single precision computational-1 items within the dependent group specified by the immediately preceding 61 DEPENDENT: entry.
6. Phantom pointers can be defined only for source IDS databases.
7. If within the group physically containing the reference code item, that item is zero, then no relation instance between the defined parent group and the member group will be created. Restated, all values of item-name-1, item-name-2, etc. must be either zero or a legal reference code of the parent group type.

[Phantom-pointer definition]



Original MD Section

```
01 APPLIANCE TYPE IS 1 ...
   02 APPLIANCE-NAME  PIC X(30).
   02 REF-CODE-OF-MANUF  PIC 9(8) COMP-1.
98 ...

01 MANUFACTURER TYPE IS 2 ...
   02 MANUF-NAME  PIC X(30).
   02 MANUF-CITY  PIC X(15).
98 ...
```

Required Extended MD Section

```
01 APPLIANCE TYPE IS 1 ...
   02 APPLIANCE-NAME  PIC X(30)
   02 REF-CODE-OF-MANUF  PIC 9(8) COMP-1.
   02 TRANSLATION-INFORMATION SIZE 0.
   61 P-K.
   61 G:  APPLIANCE,
   61 I:  APPLIANCE-NAME.
98 ...

01 MANUFACTURER TYPE IS 2 ...
   02 MANUF-NAME  PIC X(30).
   02 MANUF-CITY  PIC X(15).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 P-K.
   61 G:  MANUFACTURER,
   61 I:  MANUF-NAME.
   61 PHANTOM-POINTERS.
   61 RELATION:
   61  MAKES-APPLIANCES,
   61 DEPENDENT:
   61  APPLIANCE,
   61 POINTER:
   61  REF-CODE-OF-MANUF.
98 ...
```

In this basic example of phantom pointer relations, note carefully that despite APPLIANCE having the item (REF-CODE-OF-MANUF) that physically implements a relation from APPLIANCE to MANUFACTURER, the user must code the relation with owner/member positions reversed. MANUFACTURER is the owner of MAKES-APPLIANCES with APPLIANCE as member. Primary keys must be defined as usual for all groups.

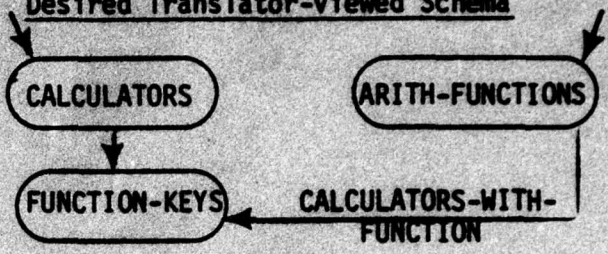
AD-A046 806 MICHIGAN UNIV ANN ARBOR GRADUATE SCHOOL OF BUSINESS--ETC F/G 9/2
DATA TRANSLATOR VERSION IIA, RELEASE 2 USER MANUAL REVISION 1.(U)
MAR 77 E KINTZER, J BODWIN, W COOL DCA100-75-C-0064
UNCLASSIFIED WP-DT-3.4 CCTC-CSM-UM-236-77-REV-1 NL

2 OF 4
AD
A046806

[Phantom pointer definition]

Original Schema

Desired Translator-viewed Schema

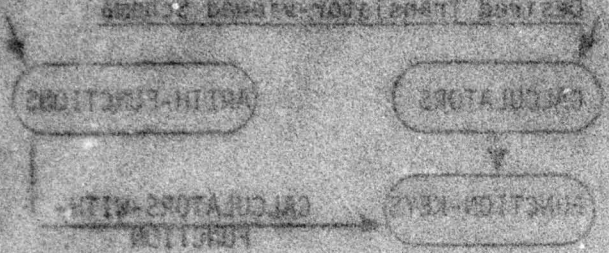


Original MD Section

```
01 CALCULATORS TYPE IS 1 ...
   02 MODEL-NAME  PIC X(16).
   02 MANUF-NAME  PIC X(20).
   02 HAS-FUNCTIONS SIZE 300.
      03 FUNCTION-KEYS OCCUR 50 TIMES
                      SIZE 300.
         04 REF-CODE-OF-ARITH-FUNCTIONS
                      PIC 9(8) COMP-1.
98 ...

01 ARITH-FUNCTIONS TYPE IS 2 ...
   02 FUNCTION-NAME  PIC X(10).
98 ...
```

Required Extended MD Section

```
01 CALCULATORS TYPE IS 1 ...
   02 MODEL-NAME  PIC X(16).
   02 MANUF-NAME  PIC X(20).
   02 HAS-FUNCTIONS SIZE 300.
      03 FUNCTION-KEYS OCCUR 50 TIMES
                      SIZE 300.
         61 OCCURS 50 TIMES.
            04 REF-CODE-OF-ARITH-FUNCTIONS
                      PIC 9(8) COMP-1.
         61 EOG.
   02 TRANSLATION-INFORMATION SIZE 0.
   61 P-K
   61 G:  CALCULATORS,
   61 I:  MODEL-NAME.
   61 G:  FUNCTION-KEYS,
   61 I: REF-CODE-OF-ARITH-FUN
   61 CTIONS.
   61 E-K-F:
   61  CALCULATORS/OWNS/FUNCTIO
   61  N-KEY
98 ...

01 ARITH-FUNCTIONS TYPE IS 2 ...
   02 FUNCTION-NAME  PIC X(10).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 P-K.
   61 G:  ARITH-FUNCTIONS,
```



```
61 I:  FUNCTION-NAME.
61 PHANTOM-POINTERS.
61 R:
61 CALCULATORS-WITH-FUNCTIO
61 N,
61 D:  FUNCTION-KEYS,
61 PO:
61 REF-CODE-OF-ARITH-FUNCTI
61 ONS.
98 ...
```

Frequently, the user implements a set of phantom pointers within a repeating group inside an 01 record. To preserve this relationship as the example shows, requires first that the contained-in-repeating group be defined within CALCULATORS, and second, that the phantom pointer relation be denoted within the owner(e.g. ARITH-FUNCTIONS) of the relation. This is consistent with the rules of phantom pointer definition by having

a) The "pointed-to" group be the owner of the relation.

b) The "having reference code" group be the member of the relation.

Note that as usual, all primary keys must be defined for every group.

3.7 ISP and Sequential Databases

It is a capability of the Data Translator to restructure source ISP and sequential databases as well as IDS databases into new target IDS databases. To do this, the following two conditions must be met:

1. The sequential or ISP database must conform to the WWDMS T-2 requirement that every record type have a field with a constant value (of no greater than 12 characters) that identifies record instances to be of a particular type. For example, an INVENTORY record would have a field whose value was always "INVENTORY".
2. A description of the source ISP or sequential database must be written as if the database were IDS, e.g., an IDS MD section must be prepared. Furthermore, this MD section must be extended or augmented with level 61 entries according to the rules and examples of Section 3 of the User Manual.

Consider condition number 2. Since the Translator requires SDDL tables describing all user databases, and since the only method by which SDDL tables can be produced is via the IDS Analyzer it follows that a description of an ISP or sequential database must be in a form usable by the IDS Analyzer. All input to the IDS Analyzer must be an extended IDS MD section.

The steps required to produce an IDS MD section are simple if the user keeps one thing in mind - an IDS MD section stripped of its IDS peculiarities (PAGE-RANGE, ASCENDING-KEYS, CHAIN-ORDER, etc.) is a method of writing the data definition of a database schema. An IDS MD section uses the concept of 01 entries to indicate records, 02 entries to indicate items and 98 entries to designate relations. Clearly an ISP or sequential database contains records, items and relations. They are different from IDS only in their physical representation. Hence in preparing an IDS MD section for an ISP or sequential database, use the following rules:

1. Indicate records with an IDS 01 entry.
2. Indicate items with IDS 02-49 entries.
3. Indicate relations with a 98 chain master entry for the parent of the relation, and a 98 chain detail for the dependent of the relation.

A few specific restrictions must also be observed to insure correct processing by the IDS Analyzer.

1. Draw a schema diagram of the ISP or sequential database. This diagram must be of a hierarchical or tree structure. Locate the top (or root) record for the tree. It must be denoted in the MD section as a CALC CHAIN DETAIL record. Choose any item within the record to be the randomize field.
2. All relations are given a chain name. The parent record of the relation must contain for that relation the following entry:

   98 chain name CHAIN MASTER CHAIN-ORDER IS AFTER.

   The dependent of a relation must have the following entry:

   98 chain name CHAIN DETAIL SELECT CURRENT MASTER.

3. All other IDS MD rules must be followed. They include the following.

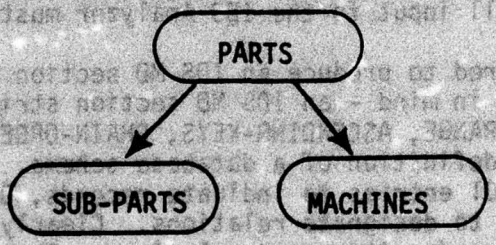
   a) all 01 records must have a TYPE IS and a RETRIEVAL VIA clause. What the user actually writes for these clauses must be acceptable to pass the IDS Translator, although the IDS Analyzer will ignore the information.

   b) all 02-49 entries must have either a SIZE clause if a group, or a PICTURE clause if an item.

Once the IDS MD section has been prepared, all of the extensions presented in the preceding part of this section in the User Manual must then be performed, if needed. Additionally, each record must have an identifier field which is prepared according to the rules of Section 3.7.1. When reading any of the rules applying to level 61 entries, the references to IDS databases should be considered as references to sequential or ISP databases.

Example

Suppose an ISP database exists with three records related as shown below. PARTS are made up of SUB-PARTS and machined on different machines.



An IDS MD section for the above ISP database would be similar to the one below.

```
01 PARTS TYPE IS 1 RETRIEVAL VIA CALC CHAIN.
   02 PARTS-ID PIC 9(1) VALUE IS 1.
   02 PART-NAME PIC X(10).
    .
    .
    .
98 HAS-SUB-PARTS CHAIN MASTER CHAIN-ORDER IS AFTER.
98 MACHINED-ON CHAIN MASTER CHAIN-ORDER IS AFTER
98 CALC CHAIN DETAIL RANDOMIZE ON PART-NAME.

01 SUB-PARTS TYPE IS 2 RETRIEVAL VIA HAS-SUB-PARTS CHAIN.
   02 SUB-PARTS-ID PIC 9  VALUE IS 2
   02 SUB-PART-NO  PIC 9(6).
    .
    .
    .
98 HAS-SUB-PARTS CHAIN DETAIL SELECT CURRENT MASTER
```

```
01 MACHINES TYPE IS 3 RETRIEVAL VIA MACHINED-ON CHAIN.
   02 MACHINES-IN PIC 9  VALUE IS 3.
   02 MACHINE-NO  PIC 9(4).
   .
   .
   .
98 MACHINED-ON CHAIN DETAIL SELECT CURRENT MASTER.
```

Note that the PARTS-ID, SUB-PARTS-ID, and MACHINES-ID fields identify their respective record types and must be followed by a 61 IDENT entry.

3.7.1 Identifier Items for ISP and Sequential Databases

As noted previously all ISP and sequential database records must contain an item that has a constant value for all instances of a given record type. This section gives the rules for describing this item via level 61 entries.

```
61 ITEM-INFORMATION.
61 IDENT:  'character string'
```

Rules

1. 'character string' is a string of up to 12 BCD characters enclosed by single quotes that is the value of the item which identifies the record for which these entries are specified. A single quote cannot be one of the BCD characters.
2. The two level 61 entries must immediately follow (in order) the 02-49 entry for the item that serves as the record identifier.
3. These entries must be specified for every record (not contained-in-repeating groups) within the ISP or sequential database being described.
4. 61 IDENT: 'characters string' cannot appear in level 61 entries for IDS databases.
5. It is assumed that 'character-string' is left-justified within the item in the record instance.

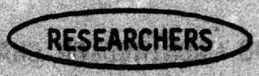
[ISP and sequential]

Original Schema

Original MD Section

```
01 RESEARCHERS TYPE IS 1 RETRIEVAL VIA
                        CALC CHAIN.
   02 RESEARCH-ID  PIC X(10).
   02 RESEARCHER-NAME  PIC X(20).
   02 RESEARCHER-PHONE  PIC 9(7).
98 CALC CHAIN DETAIL RANDOMIZE ON
     RESEARCHER-PHONE.
```

Desired Translator-viewed Schema



Required Extended MD Section

```
01 RESEARCHERS TYPE IS 1 RETRIEVAL VIA
                        CALC-CHAIN.
   02 RESEARCH-ID  PIC X(10).
   61 ITEM-INFORMATION.
   61 IDENT: 'RESEARCHER'
   02 RESEARCHER-NAME  PIC X(20)
   02 RESEARCHER-PHONE  PIC 9(7).
   02 TRANSLATION-INFORMATION SIZE 0.
   61 P-K.
   61 G:  RESEARCHERS,
   61 I:  RESEARCHER-NAME.
98 CALC CHAIN DETAIL RANDOMIZE ON
     RESEARCHER-PHONE.
```

This example is of a single record type ISP or sequential database. Every RESEARCHERS record instance possesses an item whose value is always "RESEARCHER". Note:

1. Because RESEARCHERS is at the top of the schema, it must be a CALC CHAIN DETAIL.
2. The level 61 entries for ITEM-INFORMATION and IDENT are written immediately below the RESEARCH-ID field which always has the value "RESEARCHER".
3. Primary keys must be defined as usual.

3.8 Multiple Source Databases

Another feature of the Data Translator is the ability to combine multiple source databases into multiple target databases. This section presents the technique for describing multiple databases.

The procedure for describing multiple databases is simple. Combine all MD sections for each individual database into one file. Consider this to be one large MD section of one database which is broken into sections. Combining the MD sections is necessary because only one SDDL table file is necessary to describe all source databases (similarly for target databases). Hence, only one IDS Analyzer run is required for the source database(s) descriptions, and one also for the target.

Some additional restrictions must also be observed.

1. Only 1-5 databases may be combined.
2. All chain, record and 03-49 level item names must be unique across the entire combined database description.

4.0 WRITING TDL

The Translation Definition Language (TDL) is a language in which restructuring specifications may be written in Translator-recognizable form. A complete set of restructuring specifications written in TDL is referred to as a TDL description. TDL descriptions are encoded into TDL tables by the TDL Analyzer. The Restructurer's construction of the target RIF is directed by the contents of the TDL tables.

In Section 4 is described the process by which the preliminary restructuring specifications discussed in Section 2.4 can be converted into TDL descriptions. Section 4.1 describes the TDL features needed to write TDL descriptions for any set of preliminary specifications except those involving complex item assignments and comparisons. Section 4.2 describes several advanced TDL features which can save the experienced TDL writer time and effort. It also describes the interface between the Restructurer and user-written qualification and conversion routines which are needed to perform the complex item assignments and comparisons that cannot be done directly by the Restructurer. Section 4.3 presents a complete TDL syntax summary.

4.1 Basic TDL - Tree Transcription

The central task in TDL writing is nothing more than the translation of the trees, item correspondences, and comparisons described in Section 2.4 into TDL constructs. Section 4.1.1 describes a few preliminaries used to communicate with the TDL Analyzer as well as human readers of TDL descriptions. Section 4.1.2 outlines the structure of TDL descriptions, and Sections 4.1.3-4.1.11 present the basic TDL statements.

4.1.1 Preliminaries - Comments, Toggles, Macros

Comments may appear anywhere in a TDL description. They begin with "/*" and end with "*/". For example,

```
/*  TDL TO TRANSLATE FOOTBALL DATABASE  */
```

is a valid TDL comment.

The TDL Analyzer recognizes three toggles which set and reset TDL Analyzer options. They are:

L	List (causes input lines to be echoed) default=ON.
A	Analyze (when off, allows semantic analysis of the TDL description without actually constructing TDL tables. It is useful for finding grammatical errors in a TDL description). default=ON.
D	Dump (causes TDL tables to be dumped at the end of analysis. TDL Analyzer dumps are in a user readable form. See Section 6 for details). default=OFF.

"$" sets a toggle ON, "%" turns it OFF. Toggles are set in comments. For example,

```
/*  $D %L  */
```

sets Dump ON and List OFF. This imposes the only restriction on the content of comments: if they contain an unintentional "$L", "%D", etc. surprising results may be obtained.

The TDL provides a limited, but very useful, macro facility. A macro statement has the form:

```
MACRO <NAME>  LITERALLY  <LITERAL>
```

<NAME> can be any character string that is not a TDL reserved word. <LITERAL> is any character string enclosed in single quotes. Common examples include:

```
MACRO SR LITERALLY 'SOURCE RECORD'
MACRO AV LITERALLY 'ACCESS VIA'
MACRO AT LITERALLY 'ASSIGN TO'
```

4.1.2 TDL Structure

The TDL is a block-structured language. There are four types of statements: the TARGET RECORD statement which consists of a header and one or more TDLAP statements; the TDLAP statement which consists of a header and one or more SOURCE RECORD statements; the SOURCE RECORD statement which consists of a header and one or more ITEM statements which are indivisible. This structure is shown in Figure 4-1.

```
TARGET RECORD
   TDLAP
      SOURCE RECORD
         ITEM
          .
          .
          .
      SOURCE RECORD
          .
          .
          .
   TDLAP
          .
          .
          .
TARGET RECORD
          .
          .
          .
```

Figure 4-1

TDL Block Structure

Figure 4-2 is a reproduction of the two augmented Bachman diagrams of Figure 2-13. Figure 4-3 gives a TDL description that will accomplish the transformation of 4-2a into 4-2b described by the trees of Figure 2-14, and Figure 4-4 does the same for the b-to-a restructuring of Figure 2-15. In Figure 4-3 it is assumed that the TDL description has already been tested for grammatical errors, so the List toggle has been turned off, while Dump is on so that the resulting TDL tables can be checked. The TDL description of Figure 4-4 is assumed to be freshly written, so that the A toggle has been turned off. Note the use of macros in Figure 4-4.

4.1.3 TARGET RECORD Statement

The TARGET RECORD statement is of the form:

TARGET RECORD target-record-name
[TDLAP Statement]$_1^n$

The notation $[...]_m^n$ indicates that the contents of the brackets are to occur at least m times and no more than n times. If n is not a number but a lowercase "n", then the contents of the brackets may repeat an arbitrary number of times.

There is exactly one TARGET RECORD statement for each target record type. In it, all the representations of the target record in the source database are described. The TDLAP statements which make up a TARGET RECORD statement describe the trees used to represent the target record in the source database, one TDLAP statement per tree. TARGET RECORD statements begin on lines 3, 9, and 18 of Figure 4-3, and lines 7 and 12 of Figure 4-4.

4.1.4 TDLAP Statement

The TDLAP statement is of the form:

TDLAP tdlap-name
[target-item-name = {integer / float / literal}]$_0^n$

[SOURCE RECORD Statement]$_1^n$

(Braces - {···} - indicate that exactly one of the objects inside the braces is to be chosen). A TDLAP (TDL Access Path) statement describes a tree that represents a target record type in the source RIF. There is one TDLAP statement for each of the trees developed in step 3 of Section 2.4. Each tree is assigned a unique tdlap-name, which may be one to twelve characters in length. Examples of TDLAP statements begin on lines 4,10, and 19 of Figure 4-3, and on lines 8, 13, and 20 of Figure 4-4. The trees of Figure 2-14 are described (in order of appearance) by the TDLAPs PARENT-PATH, DAUGHTERS, and SON-FINDER of Figure 4-3. The trees of Figure 2-15 are described (also in order) by the TDLAPs PARENT-PATH, DAUGHTERS, and SONS of Figure 4-4.

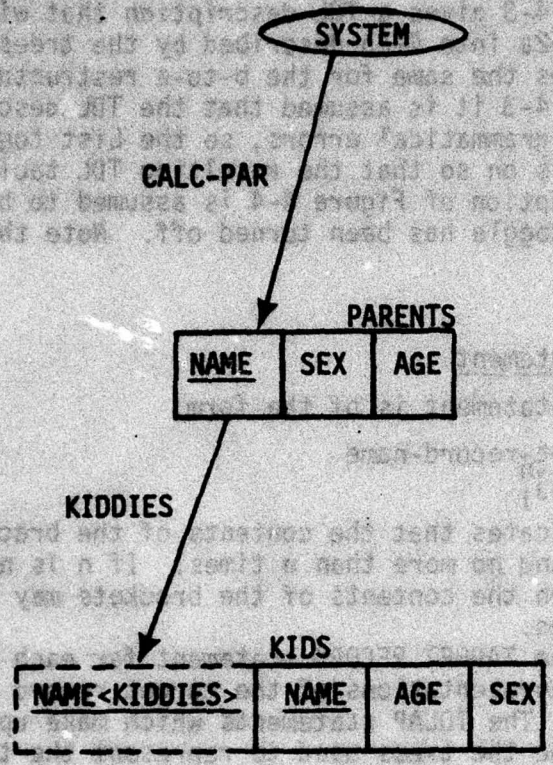


Figure 4-2a
A Database

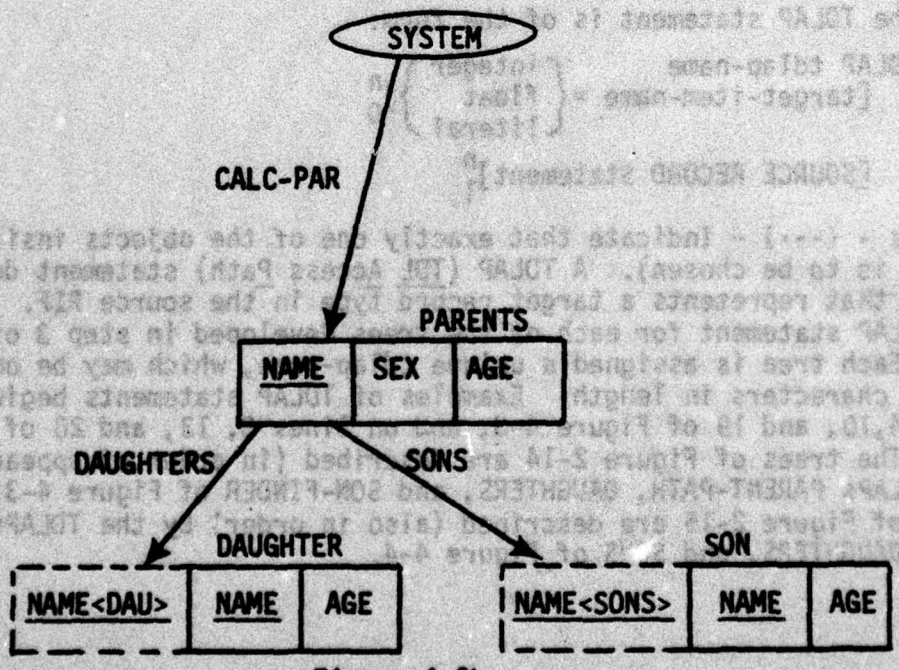


Figure 4-2b
A Related Database

```
 1.  /* TDL FOR A TO B RESTRUCTURING */
 2.  /* %L $D */
 3.  TARGET RECORD PARENTS
 4.    TDLAP PARENT-PATH
 5.      SOURCE RECORD PARENTS ACCESS VIA CALC-PAR
 6.        ACTUAL DATA IN ORDER
 7.        OTHER DATA BY NAME
 8.          /* DONE WITH PARENTS */
 9.  TARGET RECORD DAUGHTER
10.    TDLAP DAUGHTERS
11.      SOURCE RECORD PARENTS ACCESS VIA CALC-PAR
12.        NAME ASSIGN TO NAME<DAU>
13.        SOURCE RECORD KIDS ACCESS VIA KIDDIES
14.          SEX SELECT IF EQ 'FEMALE'
15.          NAME ASSIGN TO NAME
16.          AGE ASSIGN TO AGE
17.  /* END OF DAUGHTER */
18.  TARGET RECORD SON
19.    TDLAP SON-FINDER
20.      SOURCE RECORD PARENTS ACCESS VIA CALC-PAR
21.        NAME ASSIGN TO NAME<SONS>
22.        SOURCE RECORD KIDS ACCESS VIA KIDDIES
23.          SEX SELECT IF EQ 'MALE'
24.          NAME ASSIGN TO NAME
25.          AGE ASSIGN TO AGE
26.  /* END OF SON */
27.  /* TDL DESCRIPTION COMPLETE */
28.  EOF
```

Figure 4-3
TDL Example

The values of target items which are constant for all instances of the tree are specified next. Such items may be assigned integer, floating-point, or literal values, as in lines 14 and 21 of Figure 4-4. Literals must be enclosed in single quotes.

Finally, SOURCE RECORD statements are given. These statements specify the nodes of the tree, how they are to be accessed, which comparisons (if any) are to be performed on their items, and how their data is to be used to construct target record instances. Typically, SOURCE RECORD statements make up the bulk of a TDLAP statement.

4.1.5 SOURCE RECORD Statement

The SOURCE RECORD statement has the basic form:

```
SOURCE RECORD source-record-name
    ACCESS VIA source-set-name
  [ITEM Statement]n0
```

This is not the complete SOURCE RECORD statement form. Additional features are described in Sections 4.2.9 and 4.3.1.

The SOURCE RECORD statement describes a node on a tree. "source-record-name" is the name of the source record type which appears at the node, and "source-set-name" is the name of the source set which connects the node to its parent node on the tree. A parent node must be specified before any of its children can be specified. There is exactly one SOURCE RECORD statement for each node on the tree. Examples can be found at lines 5, 11, 13, 20, and 22 of Figure 4-3, and lines 9, 15, 17, 22, and 24 of Figure 4-4.

The ITEM statement takes three forms: item assignment, comparison specification, and both assignment and comparison together. These forms are described in the next three sections.

4.1.6 Item Assignments

The item assignment statement specifies the correspondence between a source item on a tree and the target item it represents. The statement has the form:

```
source-item-name ASSIGN TO target-item-name
```

Examples are lines 12, 15, 16, 21, 24, and 25 of Figure 4-3, and 16, 18 19, 23, 25, and 26 of Figure 4-4. This statement is adequate for target items which are exactly equal to source items on the tree, and those which have the same value as their source counterparts, but different RIF data types. No additional effort is required to accomplish these "implicit" conversions, since each item's RIF data type (integer, floating-point, or character) and pad character are known to the Translator. A list of these conversions appears in Figure 4-5. Note that an integer-to-character or floating-point-to-character conversion results in a right-justified character string, while in changing a character item's length or pad character, the item is assumed to be left-justified.

```
1.  /* TDL DESCRIPTION FOR B TO A RESTRUCTURING */
2.  /* %A */
3   MACRO TR LITERALLY 'TARGET RECORD'
4.  MACRO SR LITERALLY 'SOURCE RECORD'
5.  MACRO AV LITERALLY 'ACCESS VIA'
6.  MACRO IS LITERALLY 'ASSIGN TO'
7.  TR PARENTS
8.    TDLAP PARENT-PATH
9.      SR PARENTS AV CALL-PAR
10.       ACTUAL DATA IN ORDER
11.       OTHER DATA BY NAME
12. TR KIDS
13.   TDLAP DAUGHTERS
14.     SEX = 'FEMALE'
15.     SR PARENTS AV CALC-PAR
16.       NAME IS NAME<KIDDIES>
17.       SR DAUGHTER AV DAUGHTERS
18.         NAME IS NAME
19.         AGE IS AGE
20. TDLAP SONS
21.   SEX = 'MALE'
22.   SR PARENTS AV CALC-PAR
23.     NAME IS NAME<KIDDIES>
24.     SR SON AV SONS
25.       NAME IS NAME
26.       AGE IS AGE
27. /* ALL DONE*/
28. EOF
```

Figure 4-4
More TDL

Source Item Type	Target Item Type	Restrictions
1. integer	floating point	none
2. integer	character	Target item must be at least twelve characters in length. Result is right-justified in the first twelve characters of the target item and padded on the left with blanks
3. floating-point	integer	none
4. floating-point	character	same as 2
5. character	character	Source item assumed left-justified. Target item may have different length and/or pad character
6. character	integer	Item must begin with number, or be padded on left with blanks.
7. character	floating point	same as 6

Figure 4-5
"Implicit" Conversions

All other functions which convert source items to target items must be performed by user-written conversion routines. These are described in Section 4.2.3.

It should be observed that set-significant items are assigned in exactly the same way as actual data items (lines 12 and 21 of Figure 4-3 and lines 16 and 23 of Figure 4-4).

Finally, under certain circumstances, it is possible to specify the correspondence of a block of items with one statement. If all of a target record's actual data items are represented by items from a single source record, and furthermore, if the source and target record's items correspond one-for-one and in order, then

ACTUAL DATA IN ORDER

specifies that the source record's data is to be assigned as a block to the target record's data. The source record must have exactly the same number of items as the target record, they must correspond in order, corresponding items must have identical lengths and pad characters, and the actual primary key items of the source record must all correspond to actual primary key items in the target record, and vice versa. In summary, ACTUAL DATA IN ORDER may be used if the source and target records are identical or differ only in their set-significant items. ACTUAL DATA IN ORDER is used at line 6 of Figure 4-3 and line 10 of Figure 4-4. This feature could not have been used anywhere else in Figures 4-3 and 4-4.

Lines 7 of Figure 4-3 and 11 of Figure 4-4 appear because of an unpleasant anomaly in the TDL Analyzer implementation. If ACTUAL DATA IN ORDER is used, it must be the first item statement in the SOURCE RECORD statement, and it must be followed by at least one other item statement. In general, set-significant items will be assigned, or qualification criteria specified. However, some transformations will require the use of item statements which do not serve any purpose besides preventing the SOURCE RECORD statement from ending with ACTUAL DATA IN ORDER. OTHER DATA BY NAME is useful in this way, provided the target record has no set-significant items, as in the examples at hand. (For a discussion of more constructive uses of OTHER DATA BY NAME, see Section 4.2.2). If the target record has set-significant items as yet unassigned, then a selection criterion that will always be satisfied should be given. For example, line 11 of Figure 4-4 might be replaced by

SEX SELECT IF NE 'KING OF FRANCE'

(Selection criteria are discussed in detail in Section 4.1.7). This difficulty will be corrected in subsequent releases of the Translator.

4.1.7 Selection Criteria

Item comparisons are specified by item statements of the form:

source-item-name SELECT IF op { integer / float / literal }

op is one of the comparison operators EQ (equal to), NE (not equal to), GT (greater than), GE (greater than or equal to), LE (less than or equal to), or LT (less than). As before, literals must be enclosed in single quotes. Examples of this type of statement are lines 14 and 23 of Figure 4-3.

The Restructurer builds a target record occurrence from an occurrence of a tree only if all of the selection criteria for all the nodes on the tree are satisfied. Thus, in our example, DAUGHTER records will be built only from IDS records whose SEX item is equal to FEMALE, and SON records will be built only from IDS records whose SEX item contains the character string MALE followed by two blanks. No target records will be built from KIDS records with SEX items equal to 'GIRL', 'BOY', 'MALE**', 'F ', 'M ', etc. If such records exist in the source database, the information they represent will be lost in the target database. Thus, it is essential to verify that every desired target record occurrence exists in an occurrence of a tree in the source RIF that satisfies all of the selection criteria specified for that tree.

4.1.8 Item Assignments and Comparisons Together

Sometimes a restructuring transformation will require that a selection criterion be applied to a source item value, and if the comparison succeeds, the item is assigned to a target item. In that case, both the selection criterion and the item correspondence may be specified in an item statement of the form:

source-item-name SELECT IF op{integer / float / literal} ASSIGN TO target-item name

This form of the item statement makes it possible to describe completely a source item's role in a target record in a single statement. For example, if the IDS record type of Figure 4-2a were restricted to children age twelve and under, then lines 19 and 26 of Figure 4-4 would be replaced by:

```
AGE SELECT IF LE 12 ASSIGN TO AGE.
```

This is equivalent to, but slightly more economical than,

```
AGE SELECT IF LE 12
AGE ASSIGN TO AGE
```

4.1.9 Additional SOURCE RECORD Features

The basic form of the SOURCE RECORD statement is adequate for trees in which no source record type appears more than once, as in the previous two TDL examples. However, it is sometimes necessary and/or desirable to use more complicated trees on which one or more source record types appear at least twice. Figures 4-6 through 4-13 illustrate some of these more complex restructuring transformations. Figure 4-6 presents a slightly enlarged version of the "employment" database of Figure 2-9, and Figure 4-7 shows an enlarged version of the semantically valid portion of Figure 2-10. Figure 4-8 contains additional trees, which, along with the trees of Figure 2-16, specify the representation of the Figure 4-7 target database in the source database of Figure 4-6. Figure 4-9 is a TDL description of a restructuring transformation which takes the source schema of Figure 4-6 to the target schema of Figure 4-7. Figure 4-10 shows a small "ancestry" database, and Figure 4-11, a restructured version of it. Figure 4-12 gives trees to describe the restructuring transformation, and Figure 4-13, the corresponding TDL description. These examples are very

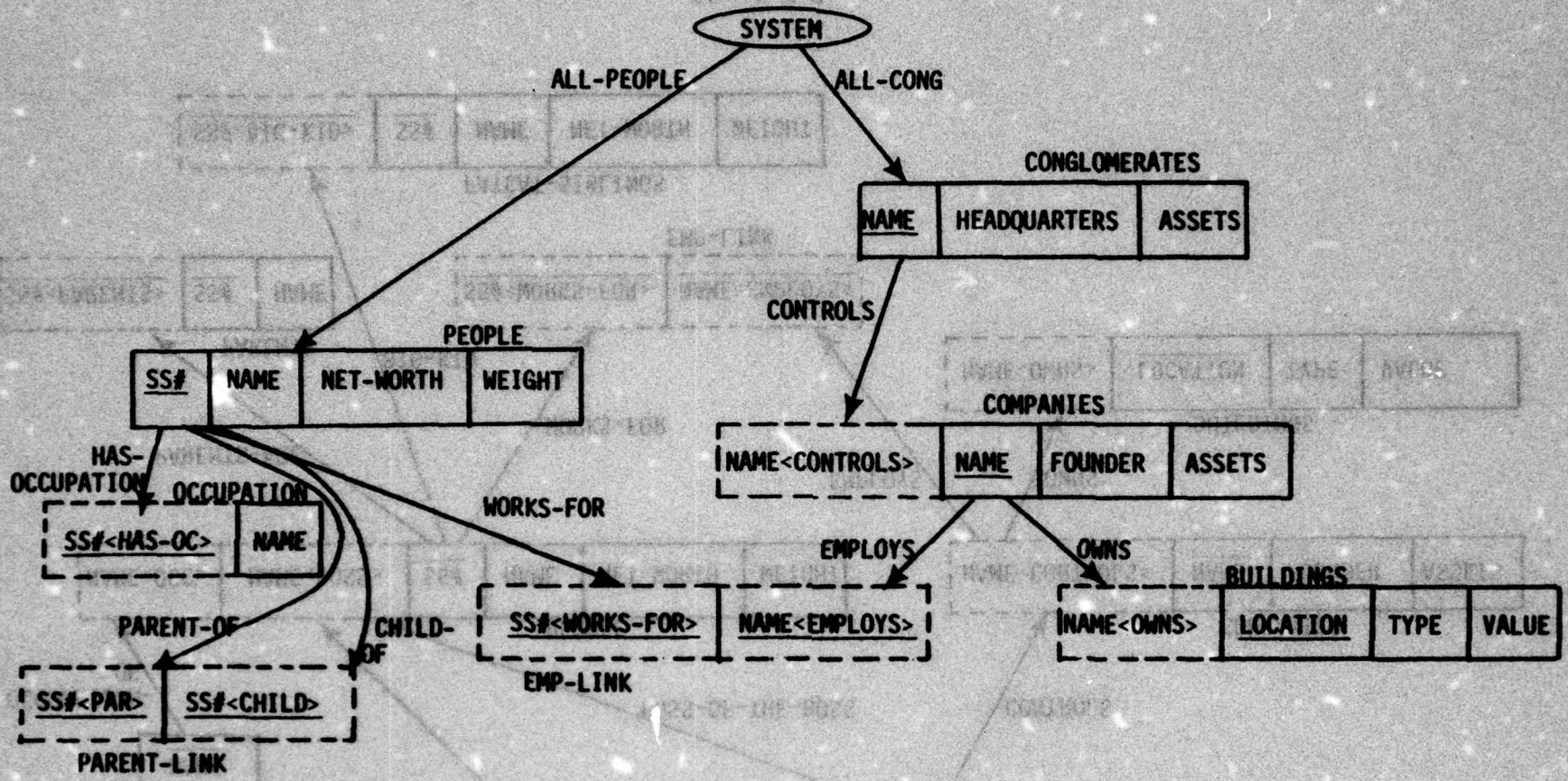


Figure 4-6
Source "Employment" Database

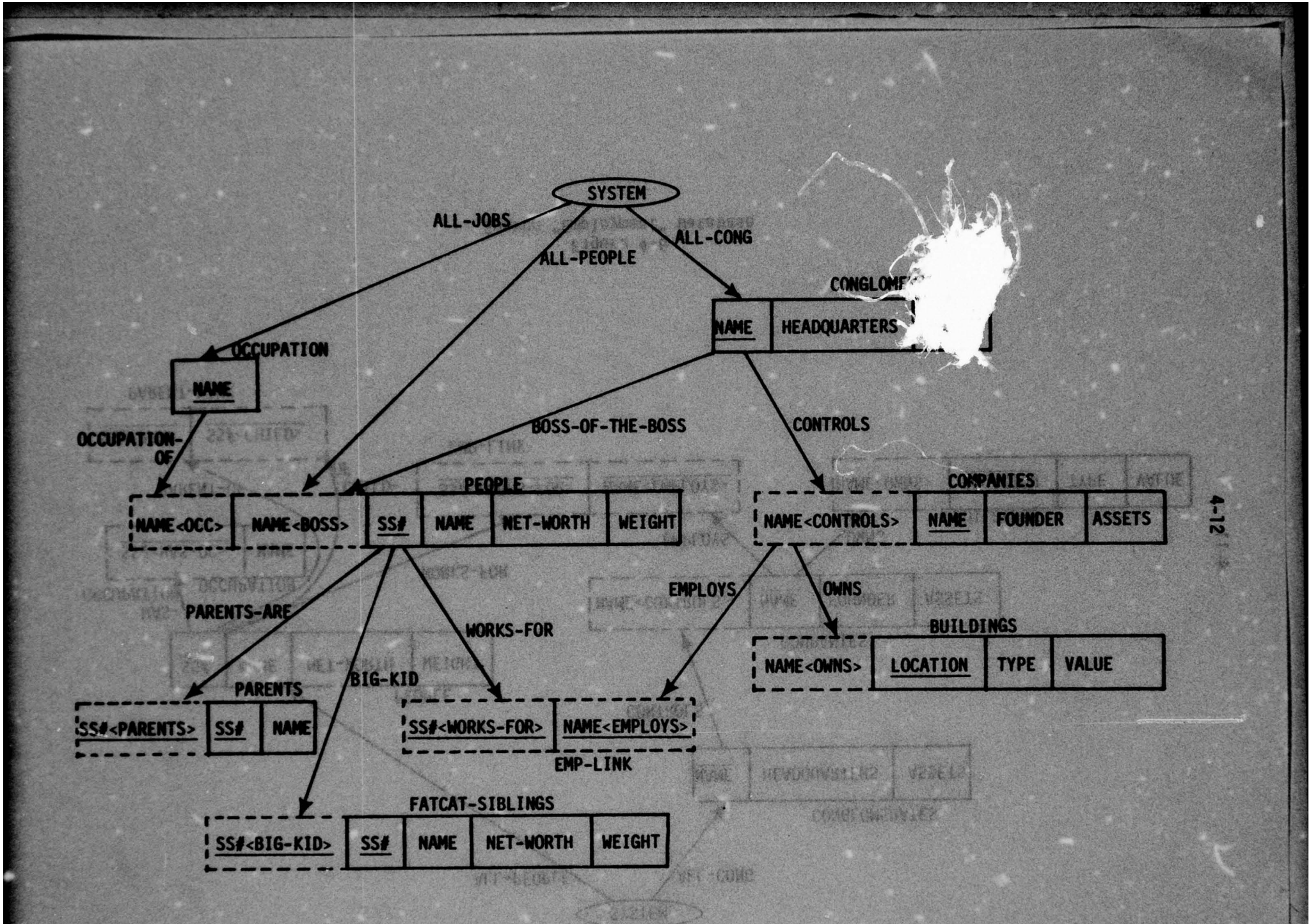


Figure 4-7
Target "Employment" Database

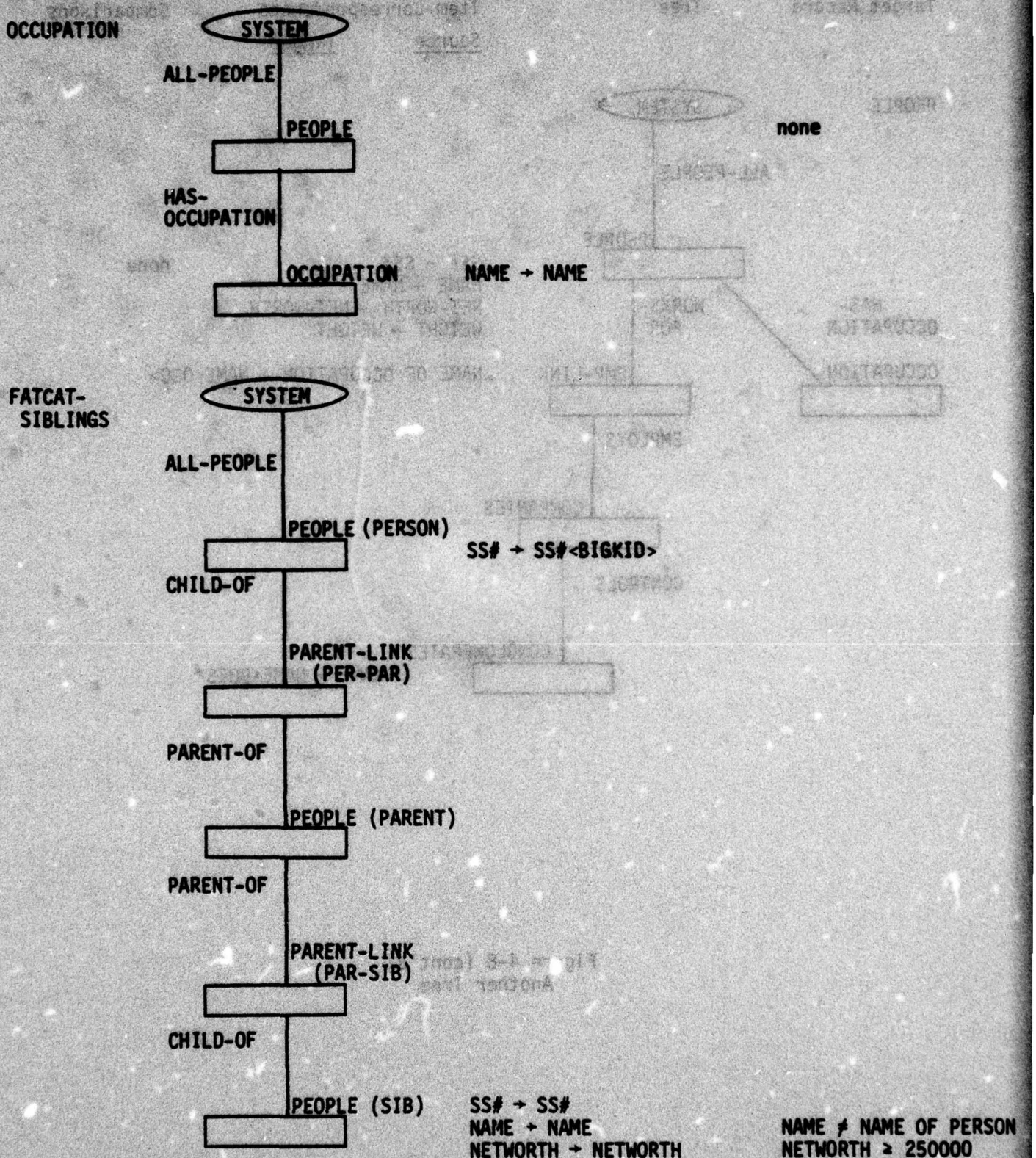


Figure 4-8
Trees

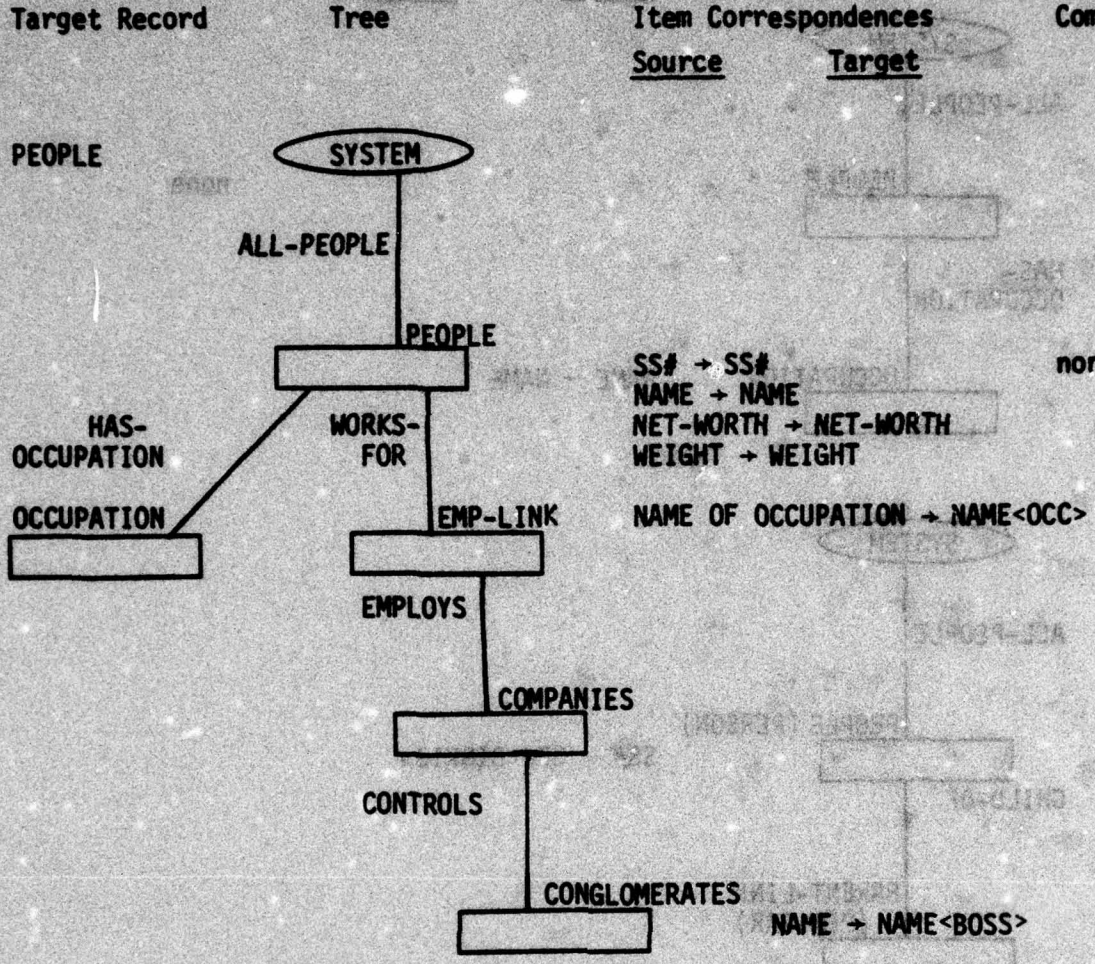


Figure 4-8 (cont'd)
Another Tree

```
1.  /*  EMPLOYMENT DATABASE TDL */
2.  MACRO SR LITERALLY 'SOURCE RECORD'
3.  MACRO AV LITERALLY 'ACCESS VIA'
4.  MACRO AT LITERALLY 'ASSIGN TO'
5.  TARGET RECORD CONGLOMERATES
6.    TDLAP CONGLOM
7.      SR CONGLOMERATES AV ALL-CONG
8.        ACTUAL DATA IN ORDER
9.        OTHER DATA BY NAME
10. TARGET RECORD COMPANIES
11.   TDLAP COMP
12.     SR CONGLOMERATES AV ALL-CONG
13.       NAME AT NAME<CONTROLS>
14.       SR COMPANIES AV CONTROLS
15.         ACTUAL DATA IN ORDER
16.         FOUNDER SELECT IF NE 'DONALD DUCK'
17. TARGET RECORD BUILDINGS
18.   TDLAP BUILD
19.     SR CONGLOMERATES AV ALL-CONG
20.       SR COMPANIES AV CONTROLS
21.         NAME AT NAME<OWNS>
22.         SR BUILDINGS AV OWNS
23.           ACTUAL DATA IN ORDER
24.           OTHER DATA BY NAME
25. TARGET RECORD EMP-LINK
26.   TDLAP EMP
27.     SR CONGLOMERATES AV ALL-CONG
```

Figure 4-9
TDL Example

```
28.         SR COMPANIES AV CONTROLS
29.           NAME AT NAME<EMPLOYS>
30.           SR EMP-LINK AV EMPLOYS
31.             SR PEOPLE AV WORKS-FOR
32.               SS# AT SS#<WORKS-FOR>
33.  TARGET RECORD PARENTS
34.    TDLAP PARENT-PATH
35.      SR PEOPLE ID=KID AV ALL-PEOPLE
36.        SS# AT SS#<PARENTS>
37         SR PARENT-LINK AV CHILD-OF
38.          SR PEOPLE ID=PARENT AV PARENT-OF
39.            SS# AT SS#
40.            NAME AT NAME
41   TARGET RECORD OCCUPATION
42.    TDLAP OCCUPATH
43.      SR PEOPLE AV ALL-PEOPLE
44.        SR OCCUPATION AV HAS-OCCUPATION
45.          NAME AT NAME
46.  TARGET RECORD FATCAT-SIBLINGS
47.    TDLAP FATCAT
48.      SR PEOPLE ID=PERSON AV ALL-PEOPLE
49.        SS# AT SS#<BIG KID>
50.        SR PARENT-LINK ID=PER-PAR AV CHILD-OF
51.          SR PEOPLE ID=PARENT AV PARENT-OF
52             SR PARENT-LINK ID=PAR-SIB AV PARENT-OF FROM ID=PARENT
53.              SR PEOPLE ID=SIB AV CHILD-OF FROM ID=PAR-SIB
54.                SS# AT SS#
55.                NAME SELECT IF NE NAME FROM PEOPLE ID=PERSON AT NAME
56.                NET-WORTH SELECT IF GE 250000 AT NET-WORTH
57.                WEIGHT SELECT IF GE 250 AT WEIGHT
```

Figure 4-9 (cont'd)

```
58.  TARGET RECORD PEOPLE
59.    TDLAP PEEP
60.      SR PEOPLE AV ALL-PEOPLE
61.        ACTUAL DATA IN ORDER
62.        WEIGHT SELECT IF GE -4000
63.        SR OCCUPATION AV HAS-OCCUPATION
64.          NAME AT NAME<OCC>
65.        SR EMP-LINK AV WORKS-FOR
66.          SR COMPANIES AV EMPLOYS
67.            SR CONGLOMERATES AV CONTROLS
68.              NAME AT NAME<BOSS>
69.  /* THE END */
70.  EOF
```

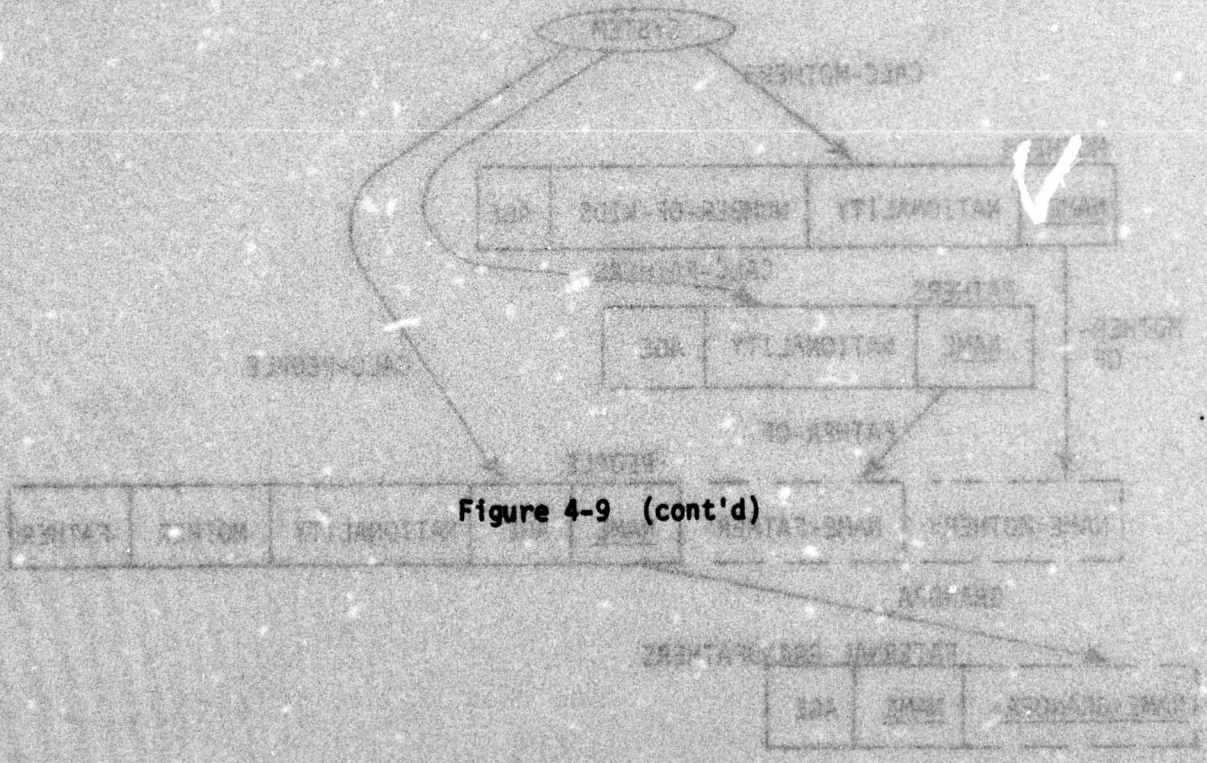


Figure 4-9 (cont'd)

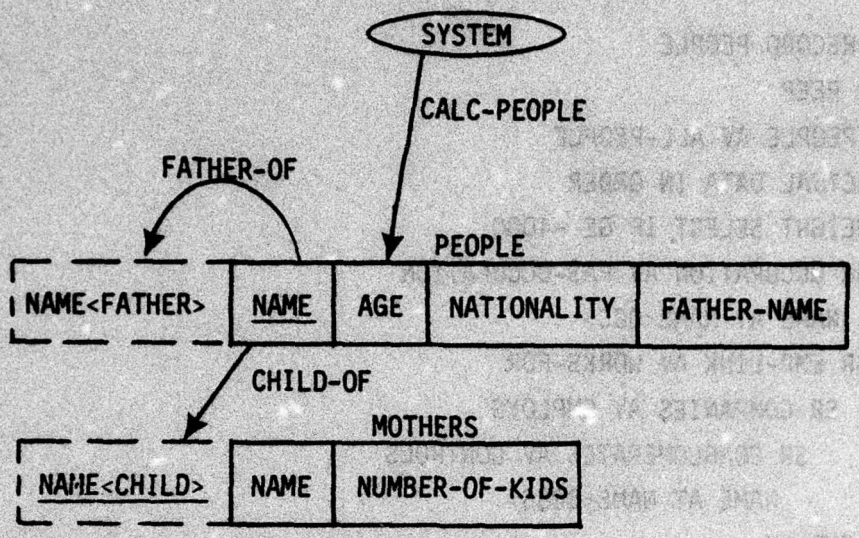


Figure 4-10
"Ancestry" Database

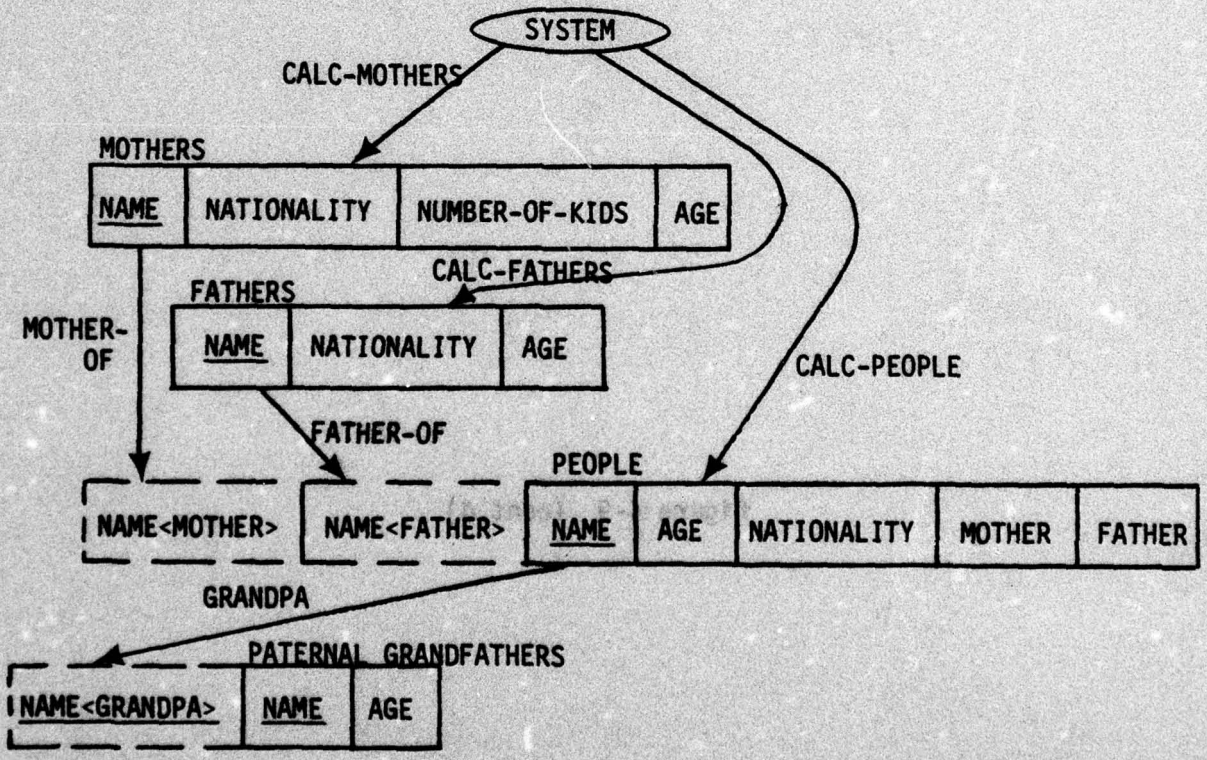


Figure 4-11
Restructured "Ancestry" Database

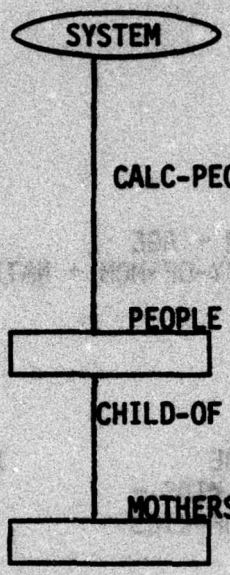
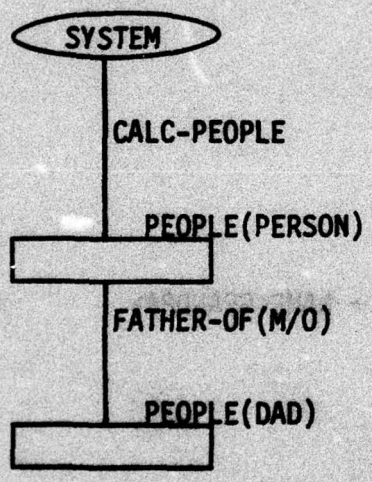
Target Record	Tree	Item Correspondences		Comparisons
		Source	Target	
PEOPLE				
		NAME	NAME	none
		AGE	AGE	
		NATIONALITY	NATIONALITY	
		FATHER-NAME	FATHER, NAME<FATHER>	
			NAME	MOTHER, NAME<MOTHER>
FATHER				
				none
		NAME	NAME	
		NATIONALITY	NATIONALITY	
		AGE	AGE	

Figure 4-12
"Family" Trees

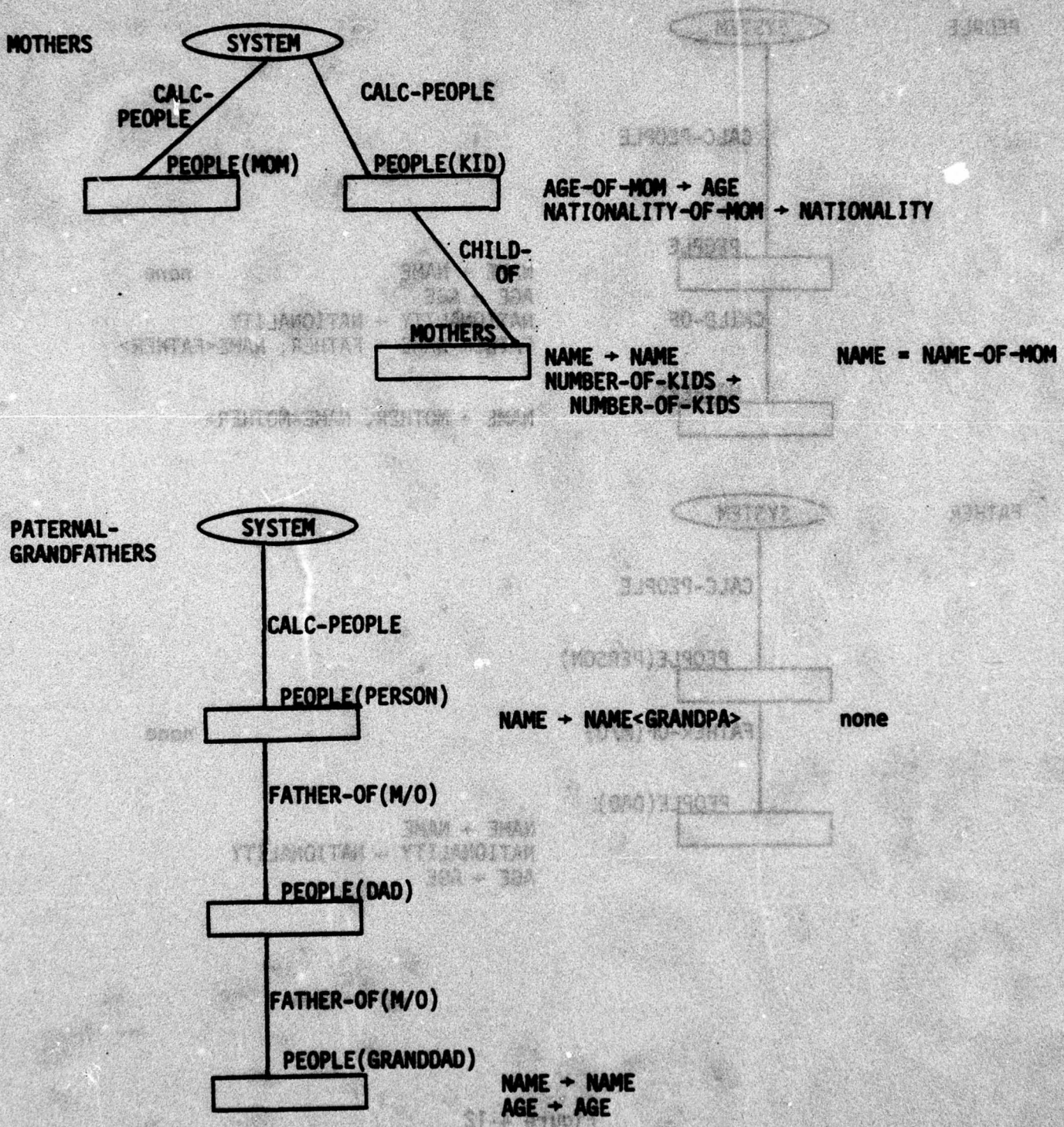


Figure 4-12 (cont'd)

```
1.  /* TDL FOR FAMILY TREE TRIMMING */
2.  MACRO TREC LITERALLY 'TARGET RECORD'
3.  MACRO SREC LITERALLY 'SOURCE RECORD'
4.  MACRO AV LITERALLY 'ACCESS VIA'
5.  MACRO IS LITERALLY 'ASSIGN TO'
6.  MACRO O/M LITERALLY 'OWNER/MEMBER'
7.  MACRO M/O LITERALLY 'MEMBER/OWNER'
8.  TREC PEOPLE
9.    TDLAP THE-PEOPLE
10.     SREC PEOPLE AV CALC-PEOPLE
11.       NAME IS NAME
12.       AGE IS AGE
13.       NATIONALITY IS NATIONALITY
14.       FATHER-NAME IS ASSIGN TO FATHER ASSIGN TO NAME<FATHER>
15.       SREC MOTHERS AV CHILD-OF
16.         NAME IS MOTHER, IS NAME<MOTHER>
17. TREC FATHERS
18.   TDLAP DADDY
19.     SREC PEOPLE ID=PERSON AV CALC-PEOPLE
20.       SREC PEOPLE ID=DAD AV FATHER-OF M/O
21.         NAME IS NAME
22.         NATIONALITY IS NATIONALITY
23.         AGE IS AGE
24. TREC MOTHERS
25.   TDLAP MAMA
26.     SREC PEOPLE ID=MOM AV CALC-PEOPLE
27.       AGE IS AGE
28.       NATIONALITY IS NATIONALITY
29.     SREC PEOPLE ID=KID AV CALC-PEOPLE
30.       SREC MOTHERS AV CHILD-OF FROM ID=KID
31.         NAME SELECT IF EQ NAME FROM PEOPLE ID=MOM IS NAME
32.         NUMBER-OF-KIDS IS NUMBER-OF-KIDS
```

Figure 4-13
TDL for Ancestry

```
33.  TREC PATERNAL-GRANDFATHERS
34.    TDLAP GRAMPS
35.      SREC PEOPLE ID=PERSON AV CALC-PEOPLE
36.        NAME IS NAME<GRANDPA>
37.        SREC PEOPLE ID=DAD AV FATHER-OF M/O
38.          SREC PEOPLE ID=GRANDDAD AV FATHER-OF FROM ID=DAD M/O
39.            NAME IS NAME
40.            AGE IS AGE
41.  /* DONE WITH ANCESTRY TDL */
42.  EOF
```

Figure 4-13 (cont'd)

contrived, of course, and they are not intended as examples of "real life" restructuring transformations. However, they are useful in illustrating a variety of TDL features.

The source database of Figure 4-6 represents the information contained in the database of Figure 2-9, and in addition, contains a WEIGHT item in the PEOPLE record, another record type, OCCUPATION, and a set, HAS-OCCUPATION. An instance of HAS-OCCUPATION consists of an instance of PEOPLE as owner, and one instance of OCCUPATION as member. The NAME item in the OCCUPATION record gives the person's current occupation. The fact that an instance of HAS-OCCUPATION can have no more than one member instance is reflected in the choice of primary key for OCCUPATION.

The target database of Figure 4-7 is an enlarged version of the target database of Figure 2-10. In addition to the restructuring described by Figure 2-16, this example inverts the relationship between PEOPLE and OCCUPATIONS, and contains a new record type, FATCAT-SIBLINGS, and a new set, BIG-KID. An instance of BIG-KID consists of a PEOPLE record as owner, and as members, one FATCAT-SIBLINGS record for each of the person's siblings who is worth at least $250,000 and weighs at least 250 pounds.

The basic TDL statements are adequate to describe the source database implementations of the target records CONGLOMERATES, COMPANIES, BUILDINGS, EMP-LINK, OCCUPATION, and PEOPLE. However, the source record type PEOPLE appears twice on the tree used to represent the PARENTS records. Each appearance has been given an identifier which distinguishes it from the others. The rule is: whenever a source record type appears in more than one SOURCE RECORD statement in a TDLAP statement, then each appearance *must be given an identifier. Identifiers must be unique* across all SOURCE RECORD statements in the TDLAP, and must be one to twelve characters in length. They are specified by:

ID = identifier

immediately following source record type names in SOURCE RECORD statements. Examples are lines 35, 38, 48,50, 51, 52, and 53 of Figure 4-9. This construction also occurs frequently in the TDL description of Figure 4-13.

The primary purpose of these identifiers is to resolve ambiguities in tree descriptions caused by multiple appearances of a source record type. Since RIF sets have one member and one owner type, specification of the source record type of node and the set which connects it to its parent node is enough to specify the source record type of the parent node. For example, line 7 of Figure 4-9 indicates that CONGLOMERATES' parent node record type is SYSTEM, line 14 indicates that COMPANIES' parent node is a CONGLOMERATES record, and line 22 indicates COMPANIES as the parent node type for BUILDINGS. If each source record type appears at most once on a TDLAP, then each SOURCE RECORD statement uniquely determines a parent node, as in the TDLAPS CONGLOM, COMP, BUILD, etc of Figure 4-9. However, when a source record type appears twice or more on a TDLAP, and it is to be the parent of some other node on the tree, then its identifier must be given in the child node's SOURCE RECORD statement in order to determine completely the parent node. This is the case in the TDLAP FATCAT of Figure 4-9, which describes FATCAT-SIBLINGS records. In lines 52 and 53,

FROM ID = identifier

immediately following the source-set-name in the SOURCE RECORD statement specifies the correct parent node. This TDL feature is also used in Figure 4-13.

Still more information, the direction of access, is required in a SOURCE RECORD statement when a source set's owner and member types are the same. This may occur when the source IDS database contains a match-key set or phantom pointer relation with the same owner and member types. In the example of Figure 4-10, FATHER-OF was a match-key set in the IDS source database, with owner match-key field NAME and member match-key field FATHER-NAME. (Notice that FATHER-NAME and NAME<FATHER> in the PEOPLE record of Figure 4-10 will always contain the same data. However, FATHER-NAME is an actual data item accessible to the user, whereas NAME<FATHER> is a set-significant item available only to the Restructurer). In line 20 of Figure 4-13, "M/O", which is a macro for "MEMBER/OWNER" is used to indicate that an instance of the parent node (PERSON) is to be a member of the FATHER-OF set owned by the child node (DAD). Thus, there can be at most one instance of the TDLAP "DADDY" containing a given instance of PEOPLE as the PERSON. On the other hand, if M/O were replaced by O/M (owner/member), then an instance of DADDY would contain a PEOPLE record instance as the PERSON and a member of its FATHER-OF set as the DAD. Thus, there would be perhaps many instances of DADDY for a given PERSON, and this would lead to a wealth of invalid FATHERS records.

In line 36 of Figure 4-13, both a FROM specification and an M/O specification are required. The complete form of the SOURCE RECORD statement is as follows:

```
SOURCE RECORD source-record-name [ID=identifier]
     ACCESS VIA source-set-name                    [(OWNER/MEMBER)]
          [FROM ID = identifier]                   [(MEMBER/OWNER)]
          [item statement]_0^n
```

4.1.10 Item vs. Item Comparisons

Some restructuring transformations require comparisons of items on a tree with other items on the tree rather than constant values. This is done in the TDLAP FATCAT of FIGURE 4-9, where a check is made (at line 55) to see that a person is not recorded as her or his own sibling, and in the TDLAP MAMA of Figure 4-13, in which line 31 is used to guarantee that target MOTHERS record occurrences are built only from source PEOPLE and MOTHERS records describing the same person.

The selection criterion syntax for item-vs-item comparisons is

```
source-item-name_1 SELECT IF op source-item-name_2
                        [FROM source-record-name [ID=identifier]]
```

The FROM specification must be used if source-item-name$_2$ does not belong to the same node as source-item-name$_1$. The source-record-name given must have previously appeared on the TDLAP. An identifier must be given if it has appeared more than once.

4.1.11 Multiple Assignments and Comparisons

In some cases, a given source item may represent more than one target item, as in the THE-PEOPLE TDLAP of Figure 4-13. As shown in lines 14 and 16 of Figure 4-13, a list of target item assignments may be used in place of a single target-item-name in an item assignment statement. The only limit on the length of such a list is the number of items in the target record.

Similarly, a source item may be required to satisfy more than one selection criterion. For example, if FATCAT-SIBLINGS were restricted to heavy people worth at least $250,000 but no more than $1,000,000, then line 56 of Figure 4-9 would be replaced by:

```
NET-WORTH SELECT IF GE 250000  SELECT IF LE 1000000
    AT NET-WORTH
```

The almost-complete syntax of an item statement is:

```
source-item-name
    [ SELECT IF  op { integer                                          }  ]n
                    { float                                            }  ]
                    { literal                                          }  ]
                    { source-item-name  [FROM source-record-name       }  ]
                                            [ID=identifier]]           }  ]o
    [ASSIGN TO    target-item-name]n
                                                                          0
```

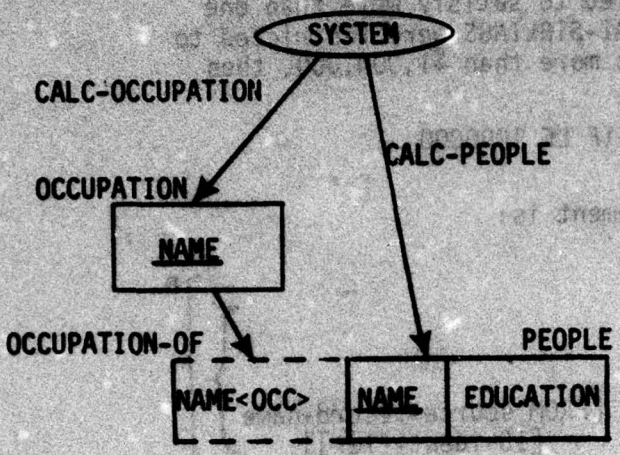
Two more constructs are available in item statements. They specify user-implemented selection criteria and conversions and are described in Section 4.2.3.

4.2 Advanced TDL Features

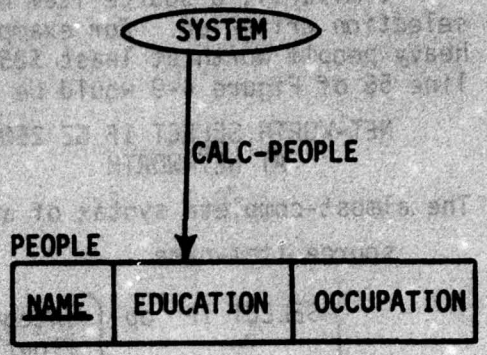
Sections 4.2.1 and 4.2.2 describe TDL features that may save the experienced TDL writer time and effort. However, they require a thorough understanding of the semantics of the source and target databases, and should be used with caution by TDL beginners. Section 4.2.3 gives specifications for user-written qualification and conversion routines. Such routines are used to perform conversion and apply selection criteria that are not handled directly by the Restucturer.

4.2.1 ACCEPT/REJECT IF NULL

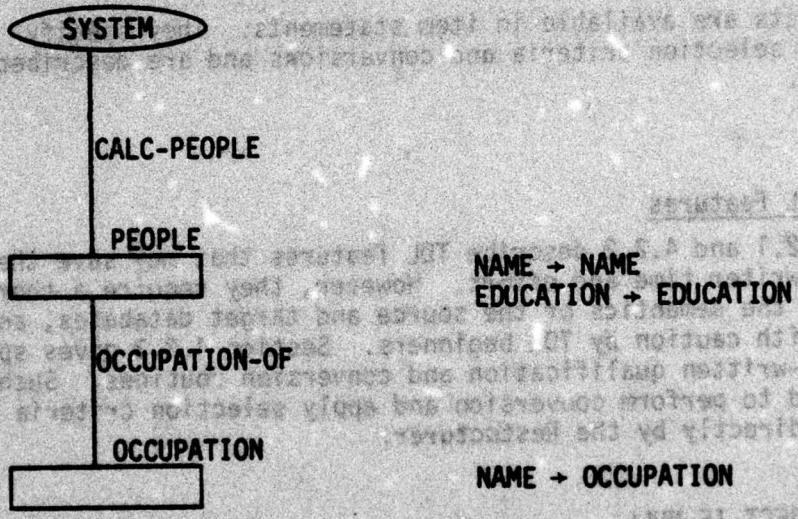
In some restructuring transformations, it is desirable to be able to create target record occurrences even when a complete instance of a TDLAP is not found. For example, consider Figure 4-14, which shows a very simple source database, target database, tree, and TDL description. Information about people whose occupation is not recorded in the database will be lost, since for them, no complete instance of PEOPLE-BUILD exists in the source data. However, it may be desirable to create target PEOPLE record occurrences for such people, and assign their OCCUPATION items the value 'UNKNOWN'. This can be accomplished by the use of ACCEPT IF NULL. The statement immediately following a SOURCE RECORD statement header may be {ACCEPT / REJECT} IF NULL. REJECT IF NULL is the default, that is, it is assumed if no such statement is given. REJECT requires that an instance of the source record be found in order to construct a target record instance.



a. Source Database



b. Target Database



c. Tree Representing Target PEOPLE

Figure 4-14
Simple Example

```
1.  TARGET RECORD PEOPLE
2.    TDLAP PEOPLE-BUILD
3.      SOURCE RECORD PEOPLE ACCESS VIA CALC-PEOPLE
4.        NAME ASSIGN TO NAME
5.        EDUCATION ASSIGN TO NAME
6.          SOURCE RECORD OCCUPATION ACCESS VIA OCCUPATION-OF
7.            NAME ASSIGN TO OCCUPATION
8.  EOF
```

d. TDL Description

Figure 4-14 (cont'd)
Simple Example

ACCEPT directs the Restructurer to construct a target record instance even if an instance of the source record is not found (i.e., the source record is null). The target items that would have been assigned values from the source record are assigned "null values" instead. Every item assignment statement of an ACCEPT IF NULL source record must specify a null value. The format is:

```
source-item-name ASSIGN TO
                                                    integer
        [target-item-name NULL VALUE = {float    }]n
                                                    literal 1
```

Figure 4-15 contains a rewritten version of the TDL description of Figure 4-14 using this feature.

Two important restrictions are imposed on ACCEPT IF NULL. First, any selection criterion which references an item belonging to a source record will fail if the source record is null, whether or not it is an ACCEPT IF NULL record. This applies to selection criteria specified for the source record, and for inter-record comparisons in which its item(s) are compared to items from other records. Secondly, and obviously, all children of an ACCEPT IF NULL node on a tree must also be ACCEPT IF NULL nodes.

```
1.  TARGET RECORD PEOPLE
2.       TDLAP PEOPLE BUILD
3.       SOURCE RECORD PEOPLE ACCESS VIA CALC-PEOPLE
4.            NAME ASSIGN TO NAME
5.            EDUCATION ASSIGN TO EDUCATION
6.            SOURCE RECORD OCCUPATION ACCESS VIA OCCUPATION-OF
7.                 ACCEPT IF NULL
8.                 NAME ASSIGN TO OCCUPATION
                        NULL VALUE = 'UNKNOWN'
9.  EOF
```

Figure 4-15

Rewritten Simple Example

As another example, suppose that in the ancestry restructuring of Figures 4-10 through 4-13, we wish to create PEOPLE record occurrences even if the mother is unknown. Then lines 15-16 of Figure 4-13 might be replaced by:

```
15          SREC MOTHERS AV CHILD-OF
15.5           ACCEPT IF NULL
16             NAME IS MOTHER NULL VALUE = 'NOT-KNOWN'
                    IS NAME<MOTHER> NULL VALUE = '     '.
```

MOTHER and NAME<MOTHER> are given different null values, since MOTHER will exist in the IDS target database where it will indicate that the mother's name is unknown whereas NAME<MOTHER> is a set-significant item which must contain a value that will not match any MOTHER record's NAME.

4.2.2 Use of Set-significant Source Items

Set-significant items in a source record may be used in selection criteria and assignments in the same way that actual data items are used, subject to one restriction: a TDLAP which uses set-significant items this way may not use ACCEPT IF NULL.

For example, Figure 4-16 shows a rewritten version of the TDLAP EMP of Figure 4-9 using source set-significant items. When source set-significant items are used, it should be remembered that they are present only if the source record instance is a member of the set to which they are significant. If not, a target record instance is not created. Thus, in the example of Figure 4-16, just as in 4-9, target EMP-LINK records are constructed only from source EMP-LINK records which belong to both the WORKS-FOR and EMPLOYS sets.

```
1.    TARGET RECORD EMP-LINK
2.         TDLAP EMP
3.              SR CONGLOMERATES AV ALL-CONG
4.                   SR COMPANIES AV CONTROLS
5.                        SR EMP-LINK AV EMPLOYS
6.                             NAME<EMPLOYS> AT  NAME<EMPLOYS>
7.                             SS#<WORKS-FOR> AT SS#<WORKS-FOR>
```

Figure 4-16

Source Set-significant items in Action

Three statements permit name-by-name assignment of blocks of items. They are:

```
SET-SIGNIFICANT DATA BY NAME
OTHER DATA BY NAME
ALL DATA BY NAME
```

SET-SIGNIFICANT DATA BY NAME is used when all of the target record's set-significant items that have not been explicitly assigned are to receive the values of set-significant items of the same name in the source record. For example, lines 6 and 7 of Figure 4-16 could be replaced by the single line:

```
6.    SET-SIGNIFICANT DATA BY NAME
```

OTHER DATA BY NAME has the same effect, and in addition, causes all of the as yet unassigned actual data items in the target record to be assigned the values of items of the same name in the source record. For example, lines 39 and 40 of Figure 4-9 could be replaced by:

```
39.    OTHER DATA BY NAME
```

Errors will result if either of these statements is used and appropriate namesakes do not exist in the source record.

Finally, ALL DATA BY NAME causes the entire target record to be assigned from items of the same name in the source record. It must be the only item assignment statement on a TDLAP. For example, lines 22-24 of example 4-9 could be replaced by:

```
21.          SR BUILDINGS AV OWNS
22.             ALL DATA BY NAME
```

The same block of statements could also be rewritten as:

```
22.  SR BUILDINGS AV OWNS
23.       ACTUAL DATA IN ORDER
24.       SET-SIGNIFICANT DATA BY NAME
```

In fact, the latter form will actually lead to a more efficient Restructurer run, since IN ORDER assignment executes more quickly than BY NAME assignment. As a rule, ACTUAL DATA IN ORDER should be used whenever possible. It often results in significant savings at Restructurer execution time.

4.2.3 User-Supplied Qualification and Conversion Routines

If it is necessary to perform more complicated item qualifications than those using only simple comparison operators (LT, GT, etc.), then a separate subroutine must be written to perform this qualification. Likewise, if an item is to be converted explicitly (i.e., not as described in Section 4.1.6), a subroutine must be written.

User-implemented selection criteria are specified in TDL item statements as follows:

source-item-name [WHEN QUALIFIED BY link-name[(VALUE STATEMENT)]]$_0^n$

where VALUE STATEMENT is of the form

{ integer
float
literal
source-item-name [FROM source-record-name [ID=identifier]] }

Link-name must be a six-character alphanumeric name and must begin with a letter. This name will be used by the Restructurer to reference the actual user-supplied subroutine. If more than one user-supplied subroutine is desired for one source item, the WHEN QUALIFIED BY clause must be repeated. A target record instance will be created using a specified source item value only if all the user-supplied subroutines indicate that it passes qualification. Also, note that SELECT IF clauses may precede the WHEN QUALIFIED BY clauses. In that case, a target record instance is created only when a) all of the comparisons in the SELECT IF clauses are true and b) all of the user-supplied subroutines specified in the WHEN QUALIFIED BY clauses indicate that the item passes qualification. See Section 4.3 for the complete item statement syntax.

Some user-implemented selection criteria will need only the source item value as input; others may compare it to a constant value or the value of some other item on the TDLAP. In the latter case, the comparison value is specified in the VALUE STATEMENT. The FROM specification must be given if the comparison value is an item that belongs to a TDLAP node different from the one in which the item statement appears, and the identifier must be given if the node has one.

For example, in the FATCAT-SIBLINGS TDLAP of Figure 4-9, selection of FATCATS might be more complex than as shown, with lines 56 and 57 rewritten as:

```
56.  NET-WORTH WHEN QUALIFIED BY HOWRCH(NAME) ASSIGN TO NET-WORTH
57.  WEIGHT WHEN QUALIFIED BY HOWFAT ASSIGN TO WEIGHT
```

It may be that WEIGHTs can be positive or negative; a positive weight is in pounds, and a negative one in kilograms. The subroutine referred to by HOWFAT would indicate that a positive weight satisfies the fatness criterion if it is at least 250, and a negative weight passes if it is less than or equal to -113.6. The subroutine referred to by HOWRCH could consult a table indicating the percentage of each person's net worth that is in negotiable form. Then a person might qualify as a FATCAT only if his negotiable net worth exceeds $100,000.

As another example, in the restructuring of Figure 4-13, it may be that NAME items in MOTHERS records contain prefixes---"DR.", "MRS.", or "MS." for example---but NAMEs in PEOPLE records do not. Then line 31 of Figure 4-13 would be replaced by:

```
31.      NAME WHEN QUALIFIED BY NAMEMP(NAME FROM PEOPLE ID=MOM)
             ASSIGN TO NAME,
```

where NAMEMP refers to a routine which strips prefixes from names before comparing them.

Routines which perform complex conversions are specified as follows:

...ASSIGN TO target-item-name [CONVERT WITH link-name]$_0^n$

For example, continuing with our ancestry example, it may be desirable to strip titles from MOTHERS' names in the target database. A routine referred to be NAMCNV could be written to accomplish this, and line 31 would read:

```
31.      NAME WHEN QUALIFIED BY NAMCMP (NAME FROM PEOPLE ID=MOM)
             ASSIGN TO NAME CONVERT WITH NAMCNV.
```

Both qualification and conversion subroutines must contain the following COMMON area through which the item values and lengths are passed.

```
COMMON/USER/BUF1,BUF2,LEN1,LEN2,IERR
```

The variables have the following meanings:

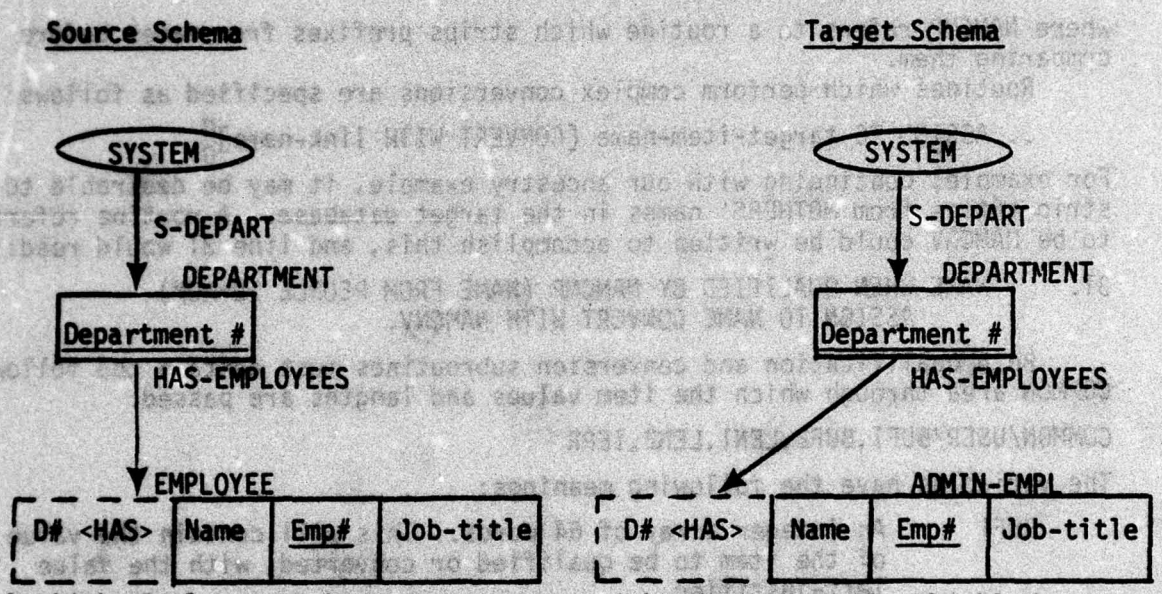
BUF1	An integer array of 64 words. This will contain the value of the item to be qualified or converted, with the value left-justified.
LEN1	An integer which contains the length of the item field in BUF1. This length is in bytes.
BUF2	An integer array of 64 words. In a qualification subroutine, this will contain the value of the item to be compared against if such a value was specified in the TDL. In a conversion operation, BUF2 will contain the converted item on exit from the subroutine. This field will be left-justified.
LEN2	In a conversion operation, an integer which contains the length of the target item in the corresponding target record. This length is also in bytes.
IERR	An integer flag set only by a qualification routine as a flag for the Restructurer: 0 - item accepted 1 - item rejected

A user-supplied subroutine must be compiled into an object file before the Restructurer is run. This object file must then be specified as a $LINK in the control card deck setup for the Restructurer. This is detailed in Section 8.7. The name given to this LINK, i.e., link-name, is the name by which the subroutine will be referred to in the TDL.

If an item is to be user-qualified, each time the Restructurer module accesses the item, it will link in the qualification subroutine and pass control to it. The item to be qualified will be in BUF1 and a second item value will be in BUF2 if one was specified in the TDL. The qualification subroutine will return a zero (0) in IERR if the item is to be accepted, and a one (1) if the item is to be rejected.

Likewise, if an item is to be user-converted, each time the Restructurer is about to assign the item to a record in the target RIF, it will link in the conversion subroutine and pass control to it. The item to be converted will be in BUF1, and the subroutine should return the converted value in BUF2. For example:

The following restructuring is to be performed:



An Admin-Empl record is to be constructed in the target DB only if the employee's job title is 'Manager' or 'Secretary'. The TDL required is:

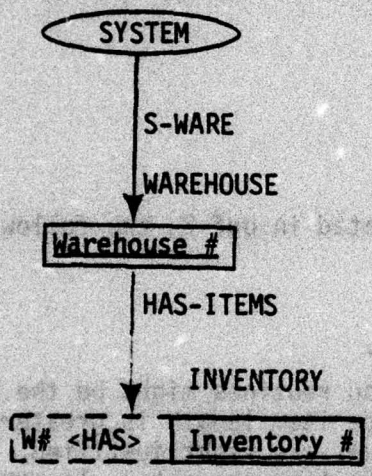
```
1.  TARGET RECORD DEPARTMENT
2.    TDLAP DEPART
3.      SOURCE RECORD DEPARTMENT ACCESS VIA S-DEPART
4.          ACTUAL DATA IN ORDER
5.          SET SIGNIFICANT DATA BY NAME
6.  TARGET RECORD ADMIN-EMPL
7.    TDLAP ADMIN
8.      SOURCE RECORD DEPARTMENT ACCESS VIA S-DEPART
9.      SOURCE RECORD EMPLOYEE ACCESS VIA HAS-EMPLOYEES
10.         JOB-TITLE WHEN QUALIFIED BY ALINK ASSIGN TO JOB-TITLE
11.         OTHER DATA BY NAME
12. EOF
```

The subroutine required by this TDL must be written and compiled into the module which has the link name ALINK in the Restructurer control cards. The subroutine needed is:

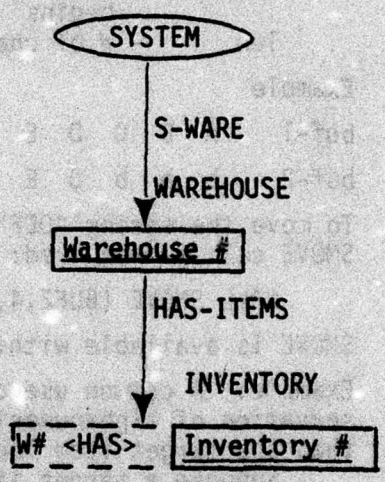
```
      SUBROUTINE USERQ1
      COMMON/USER/BUF1, BUF2, LEN1, LEN2, IERR
      INTEGER BUF1(64), BUF2(64), LEN1, LEN2, IERR
      INTEGER SECY(2), MNGR(2)
      DATA SECY/6HSECRET, 6HARY   /
      DATA MNGR/6HMANAGE, 6HR     /
      IERR=1
      IF (BUF1(1).EQ.SECY(1).AND.BUF1(2).EQ.SECY(2)
     &    .OR.BUF(1).EQ.MNGR(1).AND.BUF1(2).EQ.MNGR(2))
     &    IERR=0
      RETURN
      END
```

Example: Restructuring is to be performed where the old inventory numbers are to be suffixed with '000' to allow for more inventory numbers in the new system.

Source Schema



Target Schema



The TDL required is:

```
1.  TARGET RECORD WAREHOUSE
2.     TDLAP WAREHOUSE
3.        SOURCE RECORD WAREHOUSE ACCESS VIA S-WARE
4.           ACTUAL DATA IN ORDER
5.           SET SIGNIFICANT DATA BY NAME
6.  TARGET RECORD INVENTORY
7.     TDLAP INVENTORY
8.        SOURCE RECORD WAREHOUSE ACCESS VIA S-WARE
9.        SOURCE RECORD INVENTORY ACCESS VIA HAS-ITEMS
10.          INVENTORY# ASSIGN TO INVENTORY# CONVERT WITH BLINK
11.          SET SIGNIFICANT DATA BY NAME
12. EOF
```

The user conversion subroutine required by the above TDL must be written and compiled into the module which has the name BLINK in the Restructurer control cards. The subroutine needed is:

```
SUBROUTINE USERC1
COMMON/USER/BUF1, BUF2, LEN1, LEN2, IERR
INTEGER BUF1(64), BUF2(64), LEN1, LEN2, IERR
INTEGER SUFFIX
DATA SUFFIX/6H000    /
CALL SMOVE*(BUF2,1,BUF1,1,LEN1)
CALL SMOVE(BUF2,LEN1+1,SUFFIX,1,3)
RETURN
END
```

*SMOVE is a routine that has the following form:

CALL SMOVE (buf-1, start-1, buf-2, start-2,len)

buf-1 - address of the variable to which a string is moved

start-1 - displacement (in characters) within buf-1 where the "MOVED" string will start

buf-2 - address of the variable containing the string to move

start-2 - displacement (in characters) within buf-2 where the string begins

len - # of characters to be moved

Example

buf-1 A B C D E F G H

buf-2 b b b D E F b b

To move the string "DEF" into the position indicated in buf-2, the following SMOVE call is required:

```
CALL SMOVE (BUF2,4,BUF1,4,3)
```

SMOVE is available within the Translator library.

Example: A common use of user-supplied conversion routines might be the preservation of alphanumeric items when a) the source item length is greater than the target item length, and b) the item values are right justified.

Suppose a target item called POPULATION (data type character, length6) is to be assigned from a source item POPULATION-OF-CITY (data type character, length 10). The desired result is:

source item	target item
POPULATION-OF-CITY	POPULATION
_ _ _ _ 2 7 6 3 0 0	2 7 6 3 0 0

However, if no conversion routine is supplied, the actual result will be:

_ _ _ _ 2 7 6 3 0 0	_ _ _ _ 2 7

To avoid the unwanted truncation, a user-supplied subroutine must be used. The corresponding TDL clause might read:

```
POPULATION-OF-CITY ASSIGN TO POPULATION CONVERT WITH REJUST,
```

where REJUST refers to the following subroutine:

```
SUBROUTINE USERC2
COMMON/USER/BUF1,BUF2,LEN1,LEN2,IERR
INTEGER BUF1(64),BUF2(64),LEN1,LEN2,IERR
```

```
      INTEGER START
      START. = LEN1-LEN2 + 1
      CALL SMOVE(BUF2,1,BUF1,START,LEN2)
      RETURN
      END
```

In this case, START = 10-6+1=5; six characters will be moved from BUF1 to BUF2, the fifth through the tenth, producing the desired result. Note the use of the Translator-supplied routine SMOVE as in the previous example. The subroutine USERC2 must be compiled into an object file, which is then inserted into the Restructurer R* file in a program link called REJUST. Again, the reader is referred to Section 8.7 for details.

4.3 TDL Syntax Summary

A. TARGET RECORD STATEMENT

```
TARGET RECORD target-record-name
    [TDLAP STATEMENT]_1^n
```

Rules

1. A TDL description must contain exactly one TARGET RECORD statement for each target database record type.

B. TDLAP STATEMENT

```
TDLAP tdlap-name
    [target-item-name = {integer | float | literal}]_0^n
    [SOURCE RECORD STATEMENT]_1^n
```

Rules

1. Tdlap-name is 1-12 characters, unique over all TDLAP statements in the TDL description.
2. Each target-item-name specifies a target item which is to be assigned the specified value in each target record instance created using this access path.

C. SOURCE RECORD STATEMENT

```
SOURCE RECORD source-record-name [ID=identifier_1]
    ACCESS VIA source-set-name [FROM ID=identifier_2]
                               OWNER/MEMBER
                               MEMBER/OWNER
                [{ACCEPT | REJECT} IF NULL]
                [ACTUAL DATA IN ORDER]
                [ITEM STATEMENT]_0^n
                [BLOCK ASSIGNMENT STATEMENT]
```

Rules

1. If source-record-name a-pears in any other SOURCE RECORD statement on the TDLAP, the "ID=" clause must be used.
2. Identifier is 1-12 characters in length, and must be unique over all identifiers specified on the TDLAP.
3. source-record-name must be the owner or member record type for source-set-name.

4. If source-record-name is the owner record type for source-set-name, then at least one SOURCE RECORD statement containing the member record type for source-set-name must appear previously on the TDLAP. If source-record-name is the member record type for source-set-name, and source-set-name is not system-owned, then at least one SOURCE RECORD statement containing the owner record type for source-set-name must appear previously on the TDLAP.

5. If more than one such SOURCE RECORD statement has appeared, the "FROM" clause must be used, and $identifier_2$ must be the identifier specified for the desired parent node.

6. If source-record-name is both the owner and member record type for source-set-name, the {MEMBER/OWNER / OWNER/MEMBER} option must be used.

7. REJECT IF NULL is assumed if the {ACCEPT / REJECT} option is not used.

"ACTUAL DATA IN ORDER" Rules

1. The actual data items in the source record type must correspond 1-for-1 and in order with the actual data items in the target record type. Primary key items in the source record must correspond 1-for-1 with primary key items in the target record. No target actual data item name may appear in an item assignment statement on the TDLAP.

2. Each source actual data item must have the same length and pad character as its corresponding target actual data item.

3. ACTUAL DATA IN ORDER, when it is used, must be the first, and may not be the last, item statement in a SOURCE RECORD statement.

D. ITEM STATEMENT

source-item-name
[SELECT IF op VALUE STATEMENT]$_0^n$

[WHEN QUALIFIED BY link-name [(VALUE STATEMENT)]]$_0^n$

$$\left[\text{ASSIGN TO } \left[\text{target-item name } \left[\text{[CONVERT WITH routine-name]} \right] \left[\text{[NULL VALUE = } \left\{\begin{matrix}\text{integer}\\ \text{float}\\ \text{literal}\end{matrix}\right\} \text{]} \right] \right]_0^n \right]$$

Rules

1. See Figure 4-5, page 4-8 for implicit item conversion rules.

2. If ACCEPT IF NULL is used in any SOURCE RECORD statement on the TDLAP, source-item-name may not be the name of a set-significant item.

3. A target record instance is created if and only if all comparisons specified on the TDLAP are satisfied.

4. Results of comparisons are unpredictable if the quantities compared are of different lengths or data types.

5. NULL VALUEs must be specified for all item assignment statements when ACCEPT IF NULL was used in the SOURCE RECORD statement, and cannot be specified for items belonging to REJECT IF NULL source records.

6. See Section 4.2.3 for a detailed description of the interface to user-supplied qualification and conversion routines.

BLOCK ASSIGNMENT STATEMENTS:

SET-SIGNIFICANT DATA BY NAME
OTHER DATA BY NAME
ALL DATA BY NAME

Rules

1. SET-SIGNIFICANT DATA BY NAME is used to assign values to all the set-significant items in a target record that have not yet been assigned values. Each of the target record's remaining set-significant items is assigned the value of the source record's set-significant item of the same name (which must exist).

2. OTHER DATA BY NAME has the same effect as SET-SIGNIFICANT DATA BY NAME, except that it causes every target item not previously assigned to be given the value of the source item of the same name. This applies to both set-significant and actual data items. OTHER DATA BY NAME, when used, must be the last item assignment statement on the TDLAP.

3. ALL DATA BY NAME causes every target item to be assigned the value of the source item of the same name. When used, it must be the only item assignment statement on the TDLAP.

4. If ACCEPT IF NULL is used in <u>any</u> SOURCE RECORD statement on the TDLAP, BLOCK ASSIGNMENT statements may not be used.

E. VALUE STATEMENT

integer
float
literal
source-item-name
    [FROM source-record-name [ID=identifier]]

Rules

1. source-item-name may <u>not</u> be the name of a set-significant item if ACCEPT IF NULL is used with <u>any</u> SOURCE RECORD statement on the TDLAP.

2. The "FROM" option must be used if source-item-name does not belong to the same source record as the item to which it will be compared. Source-record-name must have appeared previously in a SOURCE RECORD statement on the TDLAP, and if it appears in more than one SOURCE RECORD statement, its identifier must be given.

5.0 RUNNING THE IDS ANALYZER

Once the user has prepared his/her source and target database extended IDS MD sections (see Section 3), it is necessary to convert these textual database descriptions into the tables used by the Data Translator. These tables, the Stored Data Definition Language tables (SDDL), contain all of the information necessary to describe the source and target databases. This process of conversion is performed by executing the IDS Analyzer twice, once for the source database(s) and similarly, once for the target database(s). As described in Section 3, the general processing flow is as follows:

1. Initialize temporary IDS database for use by IDS Translator in query mode.
2. Run IDS Translator in query mode, inputting the prepared extended IDS MD section and outputting the IDS Query dictionary.
3. Read the IDS Query dictionary as input to the IDS Analyzer and output SDDL tables and reports.

The above sequence is done twice, once for the source database(s) and again for the target database(s).

Before executing the IDS Analyzer, the following must be performed.

1. An extended IDS MD section has been written. All rules specified in Section 3 of this User Manual must have been obeyed. Furthermore, as recommended in Section 3.2.1 Rule #1, the IDS MD section prior to augmentation with level 61 entries should be compiled by the IDS Translator into a dummy COBOL-IDS program so that all IDS or COBOL errors can be removed.
2. A permanent file (random) must be created to hold each SDDL table database (one for the source, one for the target). A rule of thumb to use in creating the file is

   4 llinks/01 record

3. The first card of the extended IDS MD section must be an IDS QUERY card. "IDS QUERY" must begin exactly in column 8. All cards starting with "MD" must be removed. Hence an extended IDS MD section will look like the following prior to IDS Analyzer execution. No line within the extended MD section may extend beyond column 72. Additionally, if the extended MD section is in a file, tab characters of ":", "'", ".", ",", ";" should not be used under any circumstances.

```
col 8                                                    col 72

   IDS QUERY
   01 record ...
      02-49 plus level 61 entries
   98 entries
*
*
   01 record ...
      02-49 plus level 61 entries
   98 entries
*
*
   01 record ...
      02-49 plus level 61 entries
   08 entries
      .
      .
      . etc.
```

Figure 5-1
Extended IDS MD Section

5.1 Detailed Overview of IDS Analyzer Execution

Executing the IDS Analyzer requires four activities as shown below.

Activity	
1	Initialize IDS Query dictionary database via QUTI
2	Execute IDS Translator in query mode, which produces the IDS Query dictionary from the extended IDS MD section
3	Execute phase 1 of the IDS Analyzer which produces SDDL tables for groups, items, and relations
4	Execute phase 2 of the IDS Analyzer which completes the SDDL tables by adding set-significant items and finally produces two dumps of the SDDL tables in report form.

The entire process is illustrated in Figure 5-2. Each component is explained below.

IDS QUTI	Invoked in activity one via a $PROGRAM QUTI, this program initializes 360 database pages of a temporary IDS file, the Query dictionary.
IDS Query dictionary	A 360 page (320 words/page) temporary IDS database, initialized in activity one by QUTI and populated in activity two by the IDS Translator. The Query dictionary is an IDS database containing information on user databases (IDS, ISP or sequential). Its internal format is completely described in the IDS Data Query Installation Guide. Other relevant information on the Query dictionary is contained in Section 3.1 of this User Manual.
Extended IDS MD section	An ASCII file or card deck of the extended IDS MD section prepared according to the rules of Section 3 of this User Manual. Its layout is detailed in Figure 5-1. It may consist of as many as five individual database MD sections if multiple databases are to be described.
IDS Translator	Invoked by the $IDS card, the IDS Translator is placed into query mode by the appearance of an IDS QUERY card in the input file. Instead of producing an object form Definition Structure then passing control to COBOL, the IDS Translator in query mode is only a one-pass affair, outputting the IDS Query dictionary as a stored form version of the input extended MD section.
Phase 1 IDS Analyzer	An R* file containing the object version of the first phase of the IDS Analyzer.

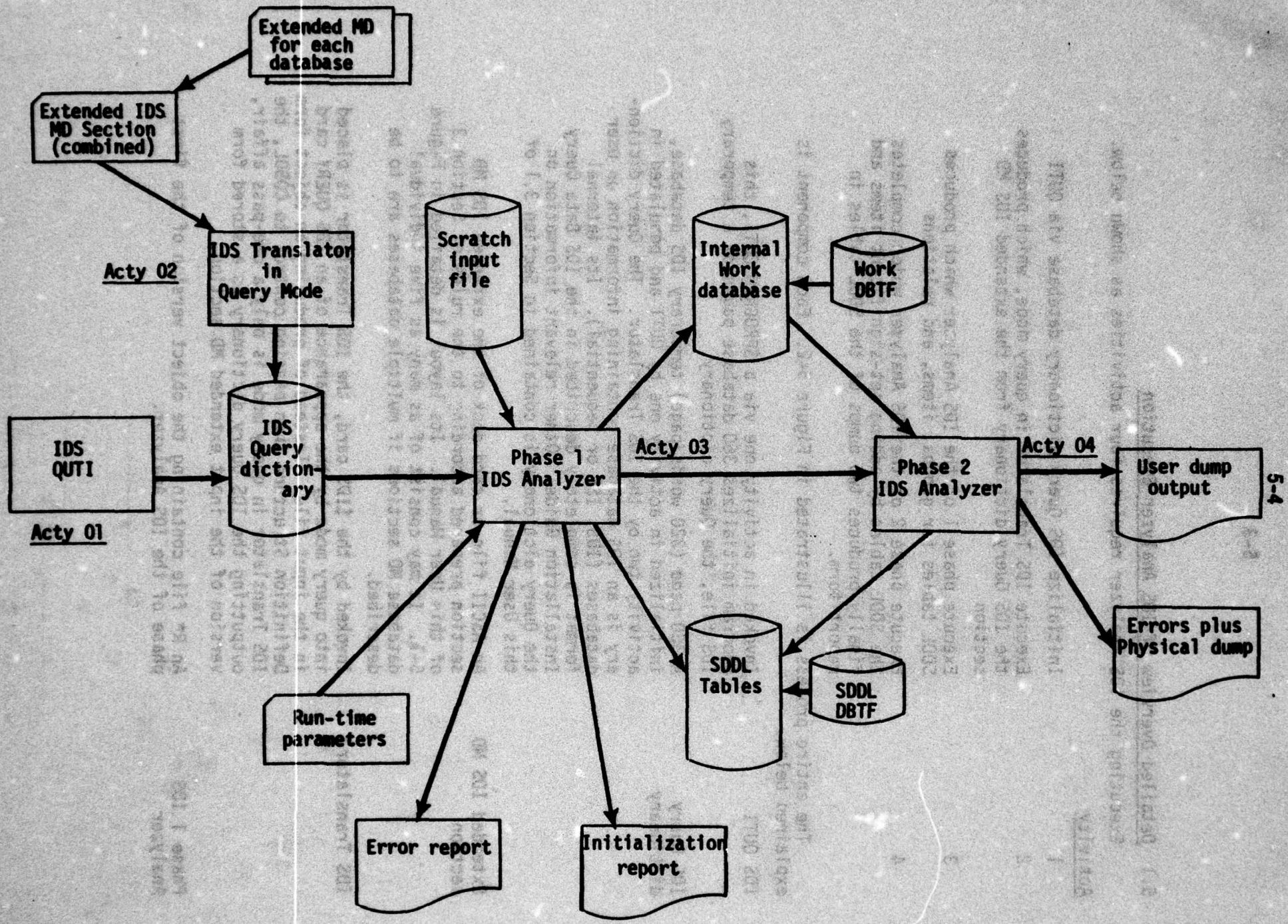


Figure 5-2
Executing the IDS Analyzer

Phase 2 IDS Analyzer	An R* file containing the object version of the second phase of the IDS Analyzer.
Runtime parameters	An ASCII file with the information on which database and records are being analyzed. See Section 5.6 for complete details.
Scratch input file	A null-linked file that must be attached for use by the ADBMS system.
SDDL tables	An ADBMS database containing the Data Translator representation of the user database(s). One SDDL tables file is sufficient for up to five source or target database descriptions. Source SDDL tables are used by the Reader, TDL Analyzer and Restructurer. Target SDDL tables are used by the TDL Analyzer, Restructurer and Writer.
SDDL DBTF	The database tables file describing to ADBMS the format and contents of the SDDL tables database. See Appendix F.
Internal Work database	A temporary ADBMS database used to communicate information between phase 1 and phase 2 of the IDS Analyzer. Once the IDS Analyzer is finished, the Internal Work database is no longer needed.
Work DBTF	The database tables file which describes to ADBMS the format and contents of the Internal Work database. See Appendix F.
Error report	A list of all syntax and semantic errors appearing while executing phase 1 of the IDS Analyzer. See Appendix A for a complete listing.
Initialization report	A report indicating that the SDDL and Internal Work databases have been initialized by ADBMS.
User dump report	Major output of the IDS Analyzer (report-wise). Contains a list of the contents of the SDDL tables. Should be used as an aid in writing TDL (see Section 4).
Errors plus physical dump	Any errors occurring during phase 2 of the IDS Analyzer are printed along with another report dumping all of the fields in the SDDL tables.

5.2 Explanation of Processing Flow

This section comprises a brief overview of the algorithms, input and outputs of the entire IDS Analyzer. Note that the whole process must be performed once for the source database(s) and again for the target database(s). Each activity is described in turn with a complete JCL description given in later sections.

5.2.1 Activity 1 - Initialize Query Dictionary

Inputs

1. IDS directives to
   a) create a temporary IDS database
   b) initialize a temporary database

Outputs

1. An initialized temporary IDS database

Algorithm

Three hundred sixty database pages have the page header record placed on them.

5.2.2 Activity 2 - Create IDS Query Dictionary

Inputs

1. Initialized temporary IDS database
2. User-prepared extended IDS MD section
3. IDS directive specifying attributes of the temporary database

Outputs

1. Complete IDS Query dictionary
2. Execution report

Algorithm

For purposes of the IDS Analyzer, the Query dictionary is created as follows:

a) 01 records are mapped to Record Definition records.

b) 02 items are mapped to Field Definition records and the immediately following Validation records.

c) 03-49 entries are mapped to Validation records which follow the Validation record for the 02 item beneath which the 03-49 entries appear.

d) level 61 entries are mapped to Description records which are linked to the Validation record for the 02-49 entry beneath which the 61 level appears.

e) 98 chain master entries are mapped to Master Definition records.

f) 98 chain detail entries are mapped to Detail Definition records.

There are other record types within the Query dictionary as well as other processing, yet as far as the IDS Analyzer is concerned, that information is ignored.

Example

Suppose that Figure 5-3 is the extended MD section. Figure 5-4 shows the Query dictionary after activity two has been executed. While it is not critical to understand completely how the Query dictionary is constructed, it is useful to be familiar with the basic Record Definition, Field Definition, Validation and Description structure so that IDS Analyzer errors can be correctly interpreted.

```
IDS QUERY
01  CITY TYPE IS 1 RETRIEVAL VIA CALC CHAIN.
    02  CITY-NAME PIC X(15).
    02  DEPARTMENTS SIZE 100.
        03  CITY-DEPTS OCCURS 10 TIMES SIZE 100.
        61  OCCURS 10 TIMES.
            04  DEPT-NAME PIC X(10).
        61  EOG.
    .
    .
    . other 02-49, 61 entries
98 CALC CHAIN DETAIL RANDOMIZE ON CITY-NAME.
98 HAS-OFFICIALS CHAIN MASTER CHAIN-ORDER IS AFTER.
01  OFFICIALS TYPE IS 2 RETRIEVAL VIA HAS-OFFICIALS CHAIN.
    .
    .
    . 02-49 + 61 entries
98  HAS-OFFICIALS CHAIN DETAIL SELECT CURRENT MASTER.
```

Figure 5-3
Sample extended IDS MD section

5.2.3 Activity 3 - Execute IDS Analyzer Phase 1

Inputs

1. IDS Query dictionary created in activity two
2. Runtime parameters
3. Scratch file
4. IDS directive specifying attributes of Query dictionary

Outputs

1. Partially complete SDDL tables
2. Internal work database
3. Initialization report
4. Error messages

Algorithm

It is important to understand the SDDL tables format in order to be able to read the user report output after activity four is complete. Figure 5-5 is the schema of the SDDL tables. Each record's contents are derived as follows:

DB	database name. All groups belonging to this database are along the GRPSIN Set. All relations within the database are found along the RELS Set.

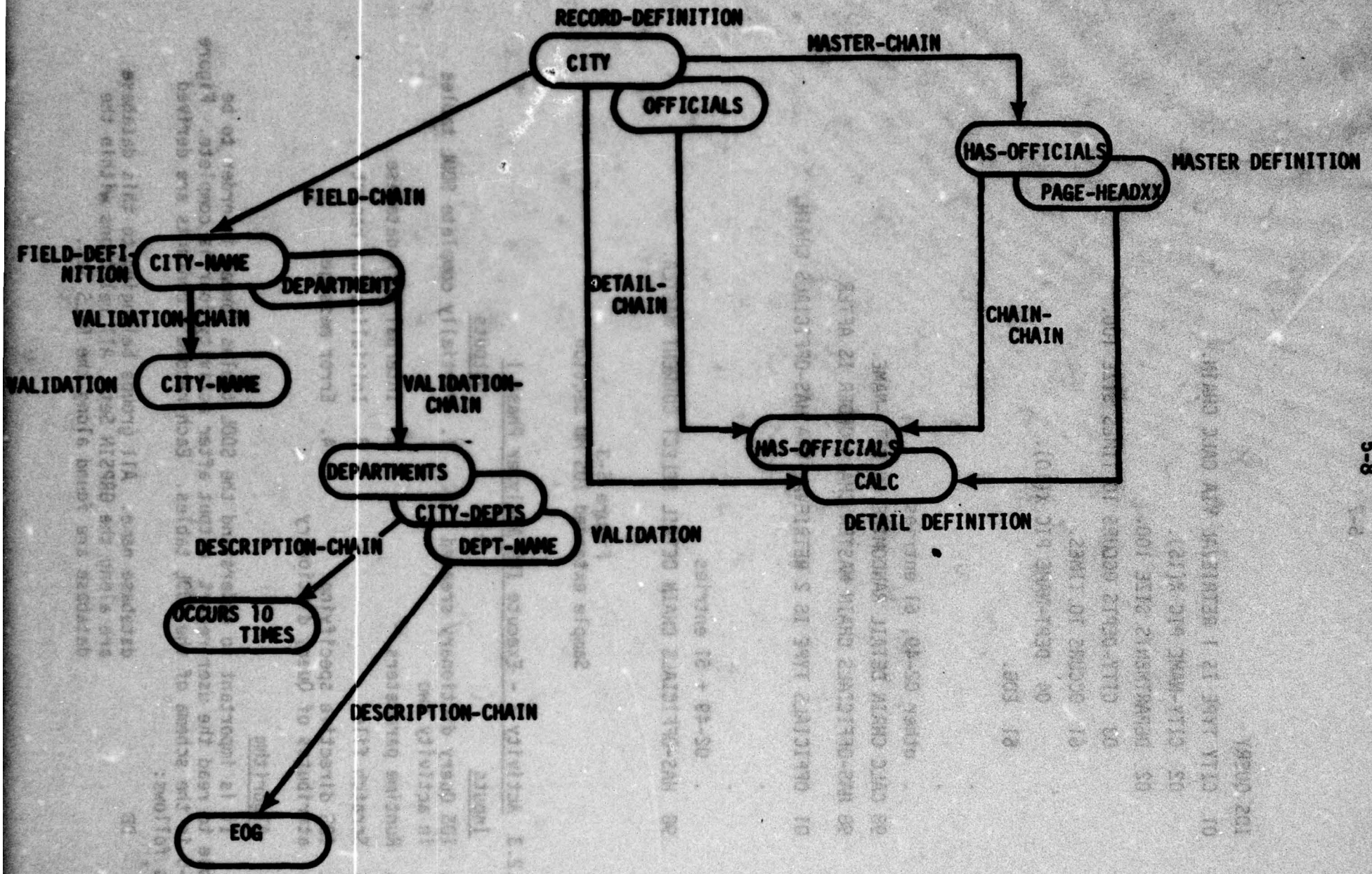


Figure 5-4
Query dictionary for MD section from Figure 5-3

GROUP	Each 01 record and contained-in-repeating group has a GROUP record for it. The relations of which the group is a member are found along the RELMEM set. The relations of which the group is an owner are found along the RELOWN set.
ITEM	Each user field is represented by an ITEM record. All the ITEMS for a GROUP are linked together on the ITEMS set. Set-significant items for a relation are linked together via the HASSIG set.
RELAY	Every relation defined by the user (chain, concatenated, match-key or phantom pointer) has its own RELAY record. The owner of the relation is found by heading the RELOWN set. The member of the relation is found by heading the RELMEM set.

The basic processing flow of the IDS Analyzer phase 1 follows.

1. Start at the first Record Definition record in the Query dictionary
   a) Create a GROUP record for the 01 record.
   b) Create RELAY records for all chains of which this record is master.
2. For each Field Definition record of the above Record Definition
   a) Create an ITEM record for the 02 entry.

      or

   i) Get the Validation record if 02 entry is for a group.
   ii) Locate the 61 OCCURS xx TIMES and create a GROUP record.
   iii) Get succeeding Validation records for the items of the contained-in-repeating group. Create ITEM records for each of these.
   b) Link the ITEM records up to the ITEMS set.
3. For the 02 TRANSLATION-INFORMATION and following level 61 entries
   a) Store primary key information in Internal Work database.
   b) Create new RELAY records for the specified phantom or match-key relations. Link sets.

The above process is repeated for every Record Definition record. While the user is certainly not expected to fully understand the details of the IDS Analyzer it is useful to have an overview of the process so that error messages can be interpreted. Many details were left out of the above description so that a basic picture can be given.

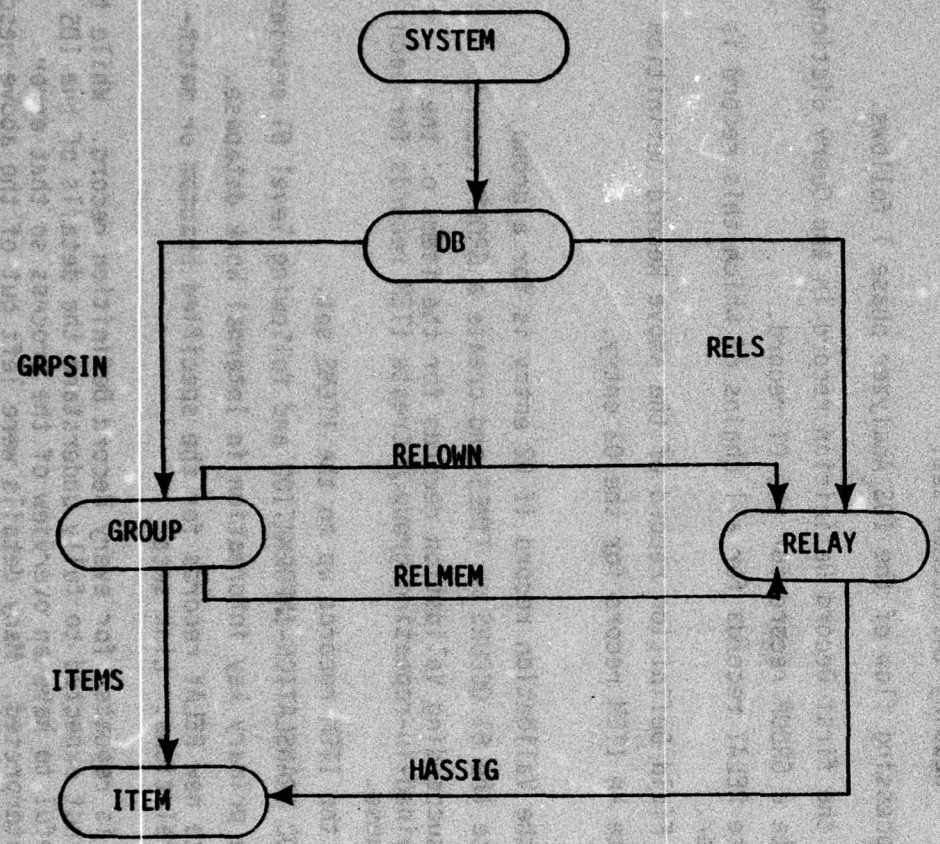


Figure 5-5
Partial SDDL table schema

5.2.4 Activity 4 - Execute IDS Analyzer Phase 2

Inputs	Outputs
1. Internal Work database	1. Complete SDDL tables
2. Partially complete SDDL tables	2. User dump of SDDL tables
	3. Error messages
	4. Physical dump of SDDL tables

Algorithm

During activity three, the IDS Analyzer collected all user-supplied information on primary keys in the Internal Work database. Phase 2 of the IDS Analyzer takes this information along with the partially complete SDDL tables (all groups, relations and user items) and creates set-significant items for the proper groups. These are then linked up to the relations (e.g. sets) for which they are significant. Once the entire SDDL tables file is complete, the two dumps are produced.

5.3 IDS Analyzer JCL

Control cards for executing all four activities are illustrated in Figure 5-6. This sequence of control cards is available on the SYSGEN tape containing the Data Translator (see Appendix G). Notes concerning the control cards are stated below where a control card is not self-explanatory.

Line no.	Remarks
0150	At this point, the user file containing the extended MD section must be placed. Note that the file must be ASCII.
0230	This is where the R* file for phase 1 of the IDS Analyzer is inserted.
0280	Supplied on the SYSGEN tape is a random library containing a variety of Translator routines used by multiple Translator Main Modules. This file must be available at load time.
0290	Attached here is the PRMFL for the SDDL tables database file. See Section 5.0 for guidelines on creation. Note that the SDDL tables file is <u>different</u> for the source and target database(s).
0300	Supplied on the SYSGEN tape is the DBTF for the SDDL tables database. It must be available at execution time.

```
0010  $       IDENT
0020  $       NOTE    ***************************************************
0030  $       NOTE    *     INITIALIZE QUERY DICTIONARY                 *
0400  $       NOTE    ***************************************************
0050  $       PROGRAM OUTI
0060  $       FILE    A1.X1S.30R                      QUERY DICTIONARY FILE
0070  $       DATA    I*
0080  IDS     INITIAL 1,360
0090  $       DATA    .Q
0100  IDS     CREATE  FC/A1/.BSSZ/360/.RNG/1.360/.INV/NO/
0110  $       NOTE    ***************************************************
0120  $       NOTE    *     CREATE IDS QUERY DICTIONARY                 *
0400  $       NOTE    ***************************************************
0140  $       IDS     NDECK
0150  $       SELECTA EXTENDED WITH LEVEL 61'S MD SECTION
0160  $       FILE    *3.X1S                          QUERY DICTIONARY
0170  $       DATA    .Q
0180  IDS     CREATE  FC/*3/.BSSZ/360/.RNG/1.360/.INV/NO/
0190  $       NOTE    ***************************************************
0200  $       NOTE    *     EXECUTE IDS ANALYZER PHASE 1                *
0400  $       NOTE    ***************************************************
0210  $       EXECUTE NREST
0220  $       LIMITS  25.64K.-2K.30000
0230  $       PRMFL   R*.R.S.PHASE 1 IDS ANALYZER R*
0240  $       FILE    H*.X2R.50R
0250  $       FILE    A1.X1R                          QUERY DICTIONARY
0260  $       DATA    .Q
0270  IDS     CREATE  FC/A1/.BSSZ/360/.RNG/1.360/.INV/NO/
0280  $       PRMFL   TR.R.R.TRANSLATOR LIBRARY
0290  $       PRMFL   03/Z1S.W.R.SDDL TABLE FILE
0300  $       PRMFL   04.P.S.SDDL DBTF
0310  $       FILE    05.X3R                          SCRATCH FILE
0320  $       REMOTE  06                              INITIALIZATION OUTPUT
0330  $       FILE    15.X4S.10R                      INTERNAL WORK DATABASE
0340  $       PRMFL   16.P.S.INTERNAL WORK DBTF
0350  $       REMOTE  A3                              ERROR MESSAGES
0360  $       DATA    A5
0370  $       SELECTA RUN-TIME PARAMETER FILE
0380  $       NOTE    ***************************************************
0390  $       NOTE    *     EXECUTE IDS ANALYZER PHASE 2                *
0400  $       NOTE    ***************************************************
0410  $       EXECUTE
0420  $       PRMFL   R*.R.S.PHASE 2 IDS ANALYZER R*
0430  $       FILE    H*.H1R.10R
0440  $       LIMITS  15.38K.-1K.15000
0450  $       PRMFL   TR.R.R.TRANSLATOR LIBRARY
0460  $       FILE    03.Z1R                          SDDL TABLES
0470  $       REMOTE  06                              SDDL DUMP AND ERROR MESSAGES
0480  $       REMOTE  07                              SDDL DUMP
0490  $       FILE    15.X4R                          INTERNAL WORK DATABASE
0500  $       ENDJOB
```

Figure 5-6
Complete IDS Analyzer Control Cards

Line no.	Remarks
0340	Supplied on the SYSGEN tape is the DBTF for the Internal Work database. It must be available at execution time.
0370	The ASCII file for the run-time parameter file (described in detail in Section 5.7) is supplied beneath the $DATA A5.
0420	See comments for line 0230.
0450	See comments for line 0280.

5.4 IDS Analyzer - Activity 1

```
0010    $       IDENT
0020    $       NOTE    ************************************************************
0030    $       NOTE    *      INITIALIZE QUERY DICTIONARY                         *
0400    $       NOTE    ************************************************************
0050    $       PROGRAM QUTI
0060    $       FILE    A1,X1S,30R                    QUERY DICTIONARY FILE
0070    $       DATA    I*
0080    IDS     INITIAL 1,360
0090    $       DATA    .Q
0100    IDS     CREATE  FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/
```

Filecode	LUD	Description
A1	X1S	A temporary random file for holding the IDS Query dictionary. Must be at least 30 random LINKS in size.
I*	--	Input to QUTI, specifies initialization directive which must be IDS INITIAL 1,360.
.Q	--	Input to IDS, indicates that a temporary IDS database is being used. Directive must be IDS CREATE FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/.

User parameters

None

Example of output

Figure 5-7 is sample output from QUTI. It is the standard IDS initialization report. Be sure to note that LLINKS ALOC=LLINKS NEC=360.

```
260T 01  10-28-76   17.327                                                    PAGE      1

IRECTIVE:  IDS    CREATE  FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/          00000000


260T 01  10-28-76   17.328                                                          F

     1 FILES ALLOCATED, RANGE      1 -   360 BASESIZE    360 BUFFERS   0

 FILCD/AREA   RANGE    PAGESIZE  PAGES/PAGE  LINES/PAGE  LLINKS ALOC  LLINKS NEC ACCESS MODE INVENTORY
    A1/ 0     1-   360   320          1 -        63          360          360       WRITE        NO


260T 01  10-28-76   17.328                                                PAGE      3

IRECTIVE:  IDS    INITIAL 1,360


260T.01  10-28-76   17.330                                                PAGE

                              I-D-S UTILIZATION REPORT

  FILE CODE          # OF READS          # OF WRITES          INVENTORY READS    INVENTORY WRITES
     A1                   0                   90                    0                   0
```

Figure 5-7
Activity 1 Output

Error output

Any occurrence of an error will be indicated on report code 74, filecode P*. Typical errors are incorrect INITIAL or CREATE directives. Appropriate action is to correct and rerun.

5.5 IDS Analyzer - Activity 2

```
0110    $       NOTE    ************************************************************
0120    $       NOTE    *     CREATE IDS QUERY DICTIONARY                          *
0400    $       NOTE    ************************************************************
0140    $       IDS     NDECK
0150    $       SELECTA EXTENDED WITH LEVEL 61 S MD SECTION
0160    $       FILE    *3,X1S                          QUERY DICTIONARY
0170    $       DATA    .Q
0180    IDS     CREATE  FC/*3/,BSSZ/360/,RNG/1,360/,INV/NO/
```

Filecode	LUD	Description
*3	X1S	The initialized IDS Query dictionary is populated by the IDS Translator in query mode in this activity. It must be attached to filecode *3.
.Q	--	All jobs that use a temporary IDS database in several activities must respecify the IDS CREATE directive for each activity in which the IDS database appears. Note that the directive is identical to the one for activity one except that FC/*3/ is used instead of FC/A1/.
S, S	--	Input to IDS Translator in query mode, e.g., the extended IDS MD section. The control card sequence illustrates a technique for inputting an ASCII file containing the extended MD section via a $SELECTA card.

User parameters

None

Example of output

The IDS Translator in query mode will produce on report code 74, file code P*, three sets of output.

a) A source listing plus errors of the extended IDS MD section.

b) A utilization report for filecode *3.

c) IDS subroutine statistics used to create the Query dictionary database.

Only the output for (a) is illustrated in Figure 5-8.

```
8260T 02  10-28-76   17.824    HIS  600  INTEGRATED DATA STORE TRANSLATOR :SR 1.

IDS ALTER NOS.

00001          01  THE-UNIVERSE TYPE IS 1
00002                RETRIEVAL VIA CALC CHAIN.
00003              02  UNIVERSE-NAME PIC X(10).
00004              02  THE-GALAXIES OCCURS 2 TIMES SIZE 48.
00005              61  OCCURS 2 TIMES.
00006                    03  MILKYWAY PIC X(5).
00007                    03  ANDROMEDA PIC X(5).
00008                    03  NUMBER OF STARS PIC 9(8) COMP-1.
00009              61  EOG.
00010              02  THE-DIPPERS PIC X(20) OCCURS 2 TIMES.
00011              61  OCCURS 2 TIMES.
00012              61  EOG.
00013              02  SOLAR-SYSTEMS  PIC X(30).
00014                  02 ASTEROIDS SIZE 72.
00015                     03 CLUMPS-OF-ROCKS OCCURS 3 TIMES SIZE 72.
00016                     61 OCCURS 3 TIMES.
00017                     61 DO-NOT-RESTRUCTURE.
00018                        04 ROCK-TYPE     PIC X(3).
00019                        04 ROCK-SPECIFIC-GRAVITY PIC 9(12) COMP-2.
00020                     61 EOG.
00021              02  PLANETS-OF-THE-SUN OCCURS 9 TIMES SIZE 90.
00022              61  OCCURS 9 TIMES.
00023              61  DO-NOT-RESTRUCTURE.
00024                    03 NAME-OF-PLANET PIC X(10).
00025              61  EOG.
00026              02  HEMISPHERES-OF-EARTH OCCURS 2 TIMES SIZE 390.
00027              61 OCCURS 2 TIMES.
00028                03  CONTINENTS PIC X(5).
00029                03  COUNTRIES-ON-CONTINENTS OCCURS 5 TIMES SIZE 190.
00030                61  OCCURS 5 TIMES.
00031                    04  STATES-IN-COUNTRIES PIC X(10).
00032                    04  CITIES-IN-STATES PIC X(10).
00033                    04  INHABITANTS-OF-CITIES OCCURS 3 TIMES SIZE 18.
00034                    61  OCCURS 3 TIMES.
00035                       05  THE-GOOD PIC X(2).
00036                       05  THE-BAD  PIC X(2).
00037                       05  THE-UGLY  PIC X(2).
00038                    61  EOG.
00039                61  EOG.
00040              61  EOG.
00041              02  TRANSLATION-INFORMATION.
***** SIZE OR PICTURE CLAUSE MISSING FROM ABOVE 02
**********************************************************************
00042              61  P-K.
00043              61  O:  THE-UNIVERSE.
00044              61  I:  UNIVERSE-NAME.
00045              61  G:  THE-GALAXIES.
00046              61  E-K-F:
00047              61 THE-UNIVERSE/OWNS/THE-GA
00048              61 LAXIES.
00049              61  O:  THE-DIPPERS.
00050              61  E-K-F:
00051              61 THE-UNIVERSE/OWNS/THE-DI
00052              61 PPERS.
```

Figure 5-8
Activity 2 Output

```
2607 02  10-28-76   17.824    HIS  600  INTEGRATED DATA STORE TRANSLATOR :SR 1/
IDS ALTER NOS.
00053                61 G: HEMISPHERES-OF-EARTH.
00054                61 E-K-F:
00055                61 THE-UNIVERSE/OWNS/HEMISP
00056                61 HERES-.
00057                61  G:
00058                61 COUNTRIES-ON-CONTINENTS.
00059                61 E-K-F:
00060                61 HEMISPHERES-/OWNS/COUNTR
00061                61 IES-ON.
00062                61  G:
00063                61 INHABITANTS-OF-CITIES.
00064                61  E-K-F:
00065                61 COUNTRIES-ON/OWNS/INHABI
00066                61 TANTS-.
00067            98  CALC CHAIN DETAIL
00068                    RANDOMIZE ON UNIVERSE-NAME.
THERE WERE  000001 WARNING FLAGS IN THE ABOVE IDS TRANSLATION
```

Figure 5-8 (cont'd)
Activity 2 Output

Error Output

Any flagrant or easily detectable syntax errors are denoted by the IDS Translator in query mode. All are self-explanatory. Common errors are:

1. REQUIRED IDS QUERY CARD MISSING

Cause: User did not include the IDS QUERY card as the first line of the extended IDS MD section. Alternatively, the rigid format of IDS Query starting in column 8 was not observed.

Action: Correct and rerun.

2. SIZE OR PICTURE CLAUSE MISSING FROM ABOVE 02

Cause: Either

a) A "SIZE 0" was not added to an 02 TRANSLATION-INFORMATION entry, or

b) An 02 COBOL group was defined without a SIZE clause. See Section 3.2.1 for rules in placing SIZE clauses.

Action: Correct and rerun.

Some errors, detectable by the IDS Translator in normal mode, cannot be detected by the IDS Translator in query mode. These errors will lead to unpredictable results. The following is a known list of errors that are not detected by the IDS Translator in query mode with the corresponding known result.

Error in extended MD Section	Result of non-detection
1. A loop (or cycle) defined with 98 CHAIK MASTER and 98 CHAIN DETAIL entries.	1. I8-RUNTIME-EXHAUSTED in activity two.
2. An elementary item whose size is larger than maximum allowable by IDS.	2. I8-RUNTIME EXHAUSTED in activity three. Impossible to detect in activity three since Query dictionary is built incorrectly.
3. A 03-49 entry that is not placed underneath an 02 entry.	3. Incorrect IDS Analyzer output. The Validation record for the errant 03-49 entry is placed underneath a Field Definition record belonging to an entirely different record type.

5.6 IDS Analyzer - Activity 3

```
0190  $     NOTE    ************************************************************
0200  $     NOTE    *      EXECUTE IDS ANALYZER PHASE 1                        *
0400  $     NOTE    ************************************************************
0210  $     EXECUTE NREST
0220  $     LIMITS  25,64 K,-2K,30000          USE APPROPIATE CORE REQ'MNT
0230  $     PRMFL   R*,R,S,PHASE 1 IDS ANALYZER R*
0240  $     FILE    H*,X2R,5OR
0250  $     FILE    A1,X1R                     QUERY DICTIONARY
0260  $     DATA    .Q
0270  IDS   CREATE  FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/
0280  $     PRMFL   TR,R,R,TRANSLATOR LIBRARY
0290  $     PRMFL   03/Z1S,W,R,SDDL TABLE FILE
0300  $     PRMFL   04,R,S,SDDL DBTF
0310  $     FILE    05,X3R                     SCRATCH FILE
0320  $     REMOTE  06                         INITIALIZATION OUTPUT
0330  $     FILE    15,X4S,1OR                 INTERNAL WORK DATABASE
0340  $     PRMFL   16,R,S,INTERNAL WORK DBTF
0350  $     REMOTE  A3                         ERROR MESSAGES
0360  $     DATA    A5
0370  $     SELECTA RUN-TIME PARAMETER FILE
```

Filecode	LUD	Description
R*	--	Phase 1 IDS Analyzer R* file from the SYSGEN tape.
H*	X2R	Required filecode so an entry can be made in PAT. Must be a random temporary file.
A1	X1R	IDS Query dictionary file initialized in activity 1 and populated in activity two. After activity three, its presence is no longer required, hence, the disposition of "R".
.Q	--	Directive to IDS describing the attributes of the temporary IDS Query dictionary database. Its format is identical to the directive in activity one.
TR	--	Random library of Data Translator object routines. Available on SYSGEN tape.
03	Z1S	A permanent random file used to contain the SDDL tables for the database description(s) processed by the IDS Analyzer. See Section 5.0 for guidelines on choosing a size.

Filecode	LUD	Description
04	--	Obtained from the SYSGEN tape, the database tables file (DBTF) for the SDDL tables database must be attached to filecode 04. See Appendix F for a description of the role of DBTFs
05	X3R	A null scratch file that ADBMS uses.
06	--	Initialization and some error output for activity three.
15	X4S	Random, temporary file for holding the Internal Work database. 10 random LINKS are sufficient for the largest extended MD section. Must have disposition of Save since it is used in activity four.
16	--	Similar to filecode 04 is the DBTF for the Internal Work database. It is available from the SYSGEN tape.
A3	--	Error message output from phase 1 of IDS Analyzer.
A5	--	Run-time parameter file (described below) is selected here into $DATA A5. It must always be available.

User parameters

For filecode A5, the user must supply a run-time parameter file with format as follows:

```
column:  1   2   3   4   5   6                    30              35
         *       dbtype-1      *   ← database name-1                →
        record-name-1 for database name-1
        record-name-2  "       "     "
              .
              .
              .
        record-name-n for      "     "
        *       dbtype-2      *   ← database name-2                →
        record-name-1 for database name-2
              .
              .
              .
```

```
column:  1   2   3   4   5   6                    30                  35
         record-name-n for database name-2
            .
            .
            .
         *     dbtype-5      *     database name-5
         record-name-1 for database name-5
            .
            .
            .
         record-name-n for database name-5
```

Rules:

1. For each database being described via one combined extended IDS MD section (see Section 3.8), the name of the database, its type and the records which belong to the database must be specified. This is necessary so the IDS Analyzer has a way to tell which 01 record within a combination of several database descriptions belongs to a given database.
2. dbtype-1 ... dbtype-5 must be either IDS, ISP, or SEQ and must be enclosed by "*" in columns 1 and 5.
3. database-name-1 ... database name-5 can be any name (up to thirty characters in length) with no preceding blanks which is assigned as the database to which the succeeding record names will belong.
4. record-name-1 ... record-name-n are names (up to thirty characters in length) which must be 01 entries within the extended IDS MD section. No preceding blanks are allowed.

Example

Suppose a user wishes to describe three source databases (one ISP, two IDS) with one combined extended MD section. Each database contains the following records:

ISP	IDS #1	IDS #2
INVENTORY	AIRCRAFT	PEOPLE
WAREHOUSE	MANUFACTURERS	ADDRESS-DATA
	SHIPS	

The run-time parameter file would then be:

```
*ISP*INVENTORY-DATA-BASE

INVENTORY
WAREHOUSE

*IDS*AIRCRAFT-SHIPS

AIRCRAFT
MANUFACTURERS
SHIPS

*IDS*PEOPLE-DATA-BASE

PEOPLE
ADDRESS-DATA
```

Examples of output

Figure 5-9 is the initialization report for the ADBMS database initializer. As noted in Appendix F of the User Manual, each ADBMS database (in this case the SDDL tables and Internal Work database) is divided into 1024 word pages and must be initialized prior to storing records in them.

Figure 5-10 is a sample of some error output produced by the IDS Analyzer. The error message is printed followed by the IDS Analyzer's current location within the IDS Query dictionary.

Error Output

See Appendix A for a complete list of errors with their causes and appropriate actions.

5.7 IDS Analyzer - Activity 4

```
)380    $       NOTE    ************************************************
)390    $       NOTE    *     EXECUTE IDS ANALYZER PHASE 2             *
)400    $       NOTE    ************************************************
)410    $       EXECUTE
)420    $       PRMFL   R*,R,S,PHASE 2 IDS ANALYZER R*
)430    $       FILE    H*,HIR,10R
)440    $       LIMITS  15,38K,-1K,15000              USE APPROPIATE CORE REQ'MNT
)450    $       PRMFL   TR,R,R,TRANSLATOR LIBRARY
)460    $       FILE    03,ZIR                        SDDL TABLES
)470    $       REMOTE  06                            SDDL DUMP AND ERROR MESSAGES
)480    $       REMOTE  07                            SDDL DUMP
)490    $       FILE    15,X4R                        INTERNAL WORK DATABASE
)500    $       ENDJOB
```

```
SNUMB = 8260T, ACTIVITY # = 03,

INITIALIZE SDDL TABLES DATABASE.

     DBIN     10/28/76  00   18.019
DATA BASE INITIALIZED WITH    37 PAGES.
INITIALIZE INTERNAL TABLES DATABASE.

     DBIN     10/28/76  00   18.022
DATA BASE INITIALIZED WITH    18 PAGES.
```

Figure 5-9
Initialization Output - Activity 3

```
SNUMB = 8260T, ACTIVITY # = 03, REPORT CODE = 03, RECORD COUNT = 000027

                    IDS ANALYZER ERROR REPORT
                          10-28-76
***ERROR...SIZE CONFLICT BETWEEN LENGTH OF GROUP AND LENGTHS OF ELEMENTARY ITEMS
          CURRENT DATA-BASE  IS= COSMIC-DATABASE
          CURRENT 01 RECORD  IS= THE-UNIVERSE
          CURRENT 02 FIELD   IS= THE-GALAXIES
          CURRENT VALIDATION IS= NUMBER
          CURRENT 61 LEVEL   IS= ECC.
***ERROR...SIZE CONFLICT BETWEEN LENGTH OF GROUP AND LENGTHS OF ELEMENTARY ITEMS
          CURRENT DATA-BASE  IS= COSMIC-DATABASE
          CURRENT 01 RECORD  IS= THE-UNIVERSE
          CURRENT 02 FIELD   IS= ASTEROIDS
          CURRENT VALIDATION IS= ROCK-SPECIFIC-GRAVITY
          CURRENT 61 LEVEL   IS= ECC.
***ERROR...SIZE CONFLICT BETWEEN LENGTH OF GROUP AND LENGTHS OF ELEMENTARY ITEM
          CURRENT DATA-BASE  IS= COSMIC-DATABASE
          CURRENT 01 RECORD  IS= THE-UNIVERSE
          CURRENT 02 FIELD   IS= ASTEROIDS
          CURRENT VALIDATION IS= ROCK-SPECIFIC-GRAVITY
          CURRENT 61 LEVEL   IS= ECC.
***ERROR...SIZE CONFLICT BETWEEN LENGTH OF GROUP AND LENGTHS OF ELEMENTARY ITEM
          CURRENT DATA-BASE  IS= COSMIC-DATABASE
          CURRENT 01 RECORD  IS= THE-UNIVERSE
          CURRENT 02 FIELD   IS= ASTEROIDS
          CURRENT VALIDATION IS= ROCK-SPECIFIC-GRAVITY
          CURRENT 61 LEVEL   IS= ECC.
```

Figure 5-10
Output Activity 3 - Report code 03

Filecode	LUD	Description
R*	--	Phase 2 IDS Analyzer R* file available from the SYSGEN tape.
H*	H1R	Filecode required so that an entry can be made in PAT. Must be a random temporary file.
TR	--	Random library of Translator object routines. Available on SYSGEN tape.
03	Z1R	SDDL tables database, partially completed in activity three.
06	--	Error report and physical dump of SDDL database.
07	--	User dump of SDDL tables database.
15	X4R	Internal Work database from activity three.

User parameters

None

Example of output

Figure 5-12 is an example physical SDDL table dump appearing on report code 06. The user need not be concerned with interpreting this output because it is used only for debugging purposes.

However, interpretation of the user SDDL table dump is critical to successful translation (see Figure 5-13 for an example). The output is organized as follows:

1. Each database is listed.
   a) for every database, the groups within it are listed.
   b) for every group, the items within the group and the relations of which the group is a master or detail are listed.
2. All relations defined are listed.
   a) for each relation, the master and detail group of the relation are printed.
3. All relations whose owner (master) is a SYSTEM are listed.

For illustrative purposes, a partial extended IDS MD section is shown in Figure 5-11. The corresponding output produced by the IDS Analyzer is shown in Figures 5-12 and 5-13. Each field of the user SDDL table dump is explained below with reference to the example extended MD section.

```
01  ADMINISTRATION TYPE IS 3 RETRIEVAL VIA CALC CHAIN.
    02  ADMIN-NUM                  PIC   X(3).
    02  INAUG-DATE SIZE 16.
        03  MONTH                  PIC   X(10).
        03  DAYE                   PIC   9(2).
        03  YEAR                   PIC   9(4).
    02  VICE-PRESIDENT SIZE 20.
        03  NAME-FIRSTN            PIC   X(10).
        03  NAME-LASTN             PIC   X(10).
    02  ADMITTED-STATES SIZE 96
        03  STATE-DREF             PIC   9(6) COMP-1  OCCURS 6 TIMES.
        61  OCCURS 6 TIMES
        61  EOG.
        03  ST-NAME                PIC   X(10)  OCCURS 6 TIMES
        61  OCCURS 6 TIMES.
        61  DO-NOT-RESTRUCTURE
        61  EOG.
    02  TRANSLATION-INFORMATION SIZE 0.
    61  PRIMARY KEYS.
    61  GROUP:
    61    ADMINISTRATION,
    61  ITEMS:  ADMIN-NUM.
    61  GROUP:
    61    STATE-DREF,
    61  E-K-F:
    61    ADMINISTRATI/OWNS/STATE-
    61    DREF.
98  CALC CHAIN DETAIL RANDOMIZE ON ADMIN-NUM.
98  STATES-ADMITTED CHAIN MASTER CHAIN-ORDER IS AFTER.
98  P-A CHAIN DETAIL SELECT CURRENT MASTER
```

Figure 5-11
Example extended MD section

```
UNIVERSITY OF MICHIGAN SDDL TABLE DUMP (VERSION IIA, RELEASE 2)

***** ALONG DBS *****

*DB      (          NAME          )      TYPE                                    KEY

*DB      OLD-PRESIDENTIAL                    0                                  000002000022

***** ALONG GRPSIN *****

*GROUP   (          NAME          )    REPEAT DBTGNM SSETNM CIRGDI CIRGLN        KEY
*ITEM    RIELEN RIETYP   PAD  SCSEC SSCTYP LENGTH DBTGNM   DISP  (  VALUE  )     KEY

*GROUP   ADMINISTRATION                      0 ADMINI SYS1.16     0      0  000002001204

*ITEM    LINE-NUMBER
*ITEM        0      0              2      0      6                 0          000006001130

*ITEM    DELETE-SWITCH
*ITEM        0      0              2      0      1                 6          000006001210

*ITEM    REC-ID
*ITEM        0      0              2      0     11                 7      3   000006001270

*ITEM    REC-SIZE
*ITEM        0      0              2      0     12                18          000006001350

*ITEM    CALC-CHAIN-NEXT
*ITEM        0      0              2      0     24                30          000006001430

*ITEM    ADMIN-NUM
*ITEM        3      1              1      6      3 ADMIN-         9          000006001510
***** OWNS THRU CRSPTO *****
          ADMIN-NUM<ADMINISTRATI/OWNS/STATE-DREF>                              000010001022
          ADMIN-NUM<STATES-ADMITTED>                                           000011000456
```

Figure 5-12
Physical dump SDDL tables

```
*ITEM     MONTHC
*ITEM        10        1                1        6       10 MONTHC      12                000006001570

*ITEM     DAYEC
*ITEM         2        1                1        6        2 DAYEC       22                000006001650

*ITEM     YEARC
*ITEM         4        1                1        6        4 YEARC       24                000006001730

*ITEM     NAME-FIRSTN
*ITEM        10        1                1        6       10 NAME-F      28                000007000016

*ITEM     NAME-LASTN
*ITEM        10        1                1        6       10 NAME-L      38                000007000076

***** CIRG *****                 STATE-DREF                                                  000006000734

*ITEM     ST-NAME
*ITEM        60        1                1        6       60 ST-NAM      87                000007000302

*ITEM     CHAIN-PTR-ITEM
*ITEM         4        3        0       2        8        4              144                000002001253

*ITEM     CHAIN-PTR-ITEM
*ITEM         4        3        0       2        8        4              148                000002001333

*ITEM     FIRSTN<P-A>
*ITEM        10        1                3        6       10 FIRST>      19                000010001652

*ITEM     INIT<P-A>
*ITEM         1        1                3        6        1 INIT<P      29                000010001732

*ITEM     LASTN<P-A>
*ITEM        10        1                3        6       10 LASTN<       9                000011000016
```

Figure 5-12 (cont'd)
Physical dump SDDL tables

```
***** KEY *****              ADMIN-NUM                                    000006001510

***** IDENT *****            REC-ID                                       000006001270

***** RELMEM *****           CALC-ADMINISTRATION                          000007000442
***** RELMEM *****           P-A                                          000002001060

***** RELOWN *****           ADMINISTRATI/OWNS/STATE-DREF                 000007000236
***** RELOWN *****           STATES-ADMITTED                              000007000362

*GROUP    STATE-DREF                        6 STATE- SYSII+     51     6 000006000734

*ITEM     STATE-DREF/IT
*ITEM        6      3      0      1      1      6 STATE-       0          000006001003

*ITEM     ADMIN-NUM<ADMINISTRATI/OWNS/STATE-DREF>
*ITEM        3      1             3      6      3 ADMIN>       9          000010001022

*ITEM     STATE-NAME<ADMITTED-BY-ADMIN>
*ITEM       10      1             3      6     10 STATE>       9          000011000376


***** KEY *****              ADMIN-NUM<ADMINISTRATI/OWNS/ST .....         000010001022

***** RELMEM *****           ADMINISTRATI/OWNS/STATE-DREF                 000007000236
***** RELMEM *****           ADMITTED-BY-ADMIN                            000006000670
```

Figure 5-12 (cont'd)
Physical dump SDDL tables

UNIVERSITY OF MICHIGAN SDDL TABLE DUMP (VERSION IIA, RELEASE 2)

***** ALONG RELS *****

```
*RELAY   (          NAME          )     DECTYP DBTGNM                          KEY
*RELMA
*RELAY   ADMINISTRATI/OWNS/STATE-DREF          2 ADMINF                  000007000236

                    ***** ALONG HASSIG *****

*ITEM    ADMIN-NUM<ADMINISTRATI/OWNS/STATE-DREF>                         000010001022

                    ***** END HASSIG *****

*RELAY   (          NAME          )     DECTYP DBTGNM                          KEY
*RELMAP  REPTYP          PT1          PT2          PT3                          KEY
*RELAY   ADMITTED-BY-ADMIN                     1 ADMITT                  000006000670
*RELMAP        0 000006001003 000000000000 000000000000 ***** MAPDEP *****  000006001050

                    ***** ALONG HASSIG *****

*ITEM    STATE-NAME<ADMITTED-BY-ADMIN>                                   000011000376

                    ***** END HASSIG *****

*RELAY   (          NAME          )     DECTYP DBTGNM                          KEY
*RELMAP  REPTYP          PT1          PT2          PT3                          KEY
*RELAY   CALC-ADMINISTRATION                   1 CALC-A                  000007000442
```

Figure 5-12 (cont'd)
Physical Dump SDDL tables

UNIVERSITY OF MICHIGAN SDDL TABLE DUMP (VERSION IIA, RELEASE 2)

***** ALONG RELAYS *****

*RELAY	(NAME)	ON SYSOWN	KEY
*RELAY	ADMINISTRATI/OWNS/STATE-DREF	----NO---	000007000236
*RELAY	ADMITTED-BY-ADMIN	----NO---	000006000670
*RELAY	CALC-ADMINISTRATION	---YES---	000007000442
*RELAY	CALC-CONGRESS	---YES---	000010000076
*RELAY	CALC-ELECTION	---YES---	000005000176
*RELAY	CALC-PRESIDENT	---YES---	000004000427
*RELAY	CALC-STATES-IN-UNION	---YE3---	000006001064
*RELAY	CONGRESS/OWNS/MAKE-UP	----NO---	000007001600
*RELAY	C-PCL	----NO---	000007001135
*RELAY	ELECTION/OWNS/ELECT-VOTES	----NO---	000005000132
*RELAY	ELECTION/OWNS/NAME-LOSER	----NO---	000004001502
*RELAY	ELECTION/OWNS/PARTY	----NO---	000004001675
*RELAY	PRESIDENT/OWNS/TITLE-OF-JOB	----NO---	000003001202
*RELAY	PRESIDENT/OWNS/W-M	----NO---	000004000003
*RELAY	P-A	----NO---	000002001060
*RELAY	P-E	----NO---	000002001443
*RELAY	P-PCL	----NO---	000002000475
*RELAY	S-P	----NO---	000005001171
*RELAY	STATES-ADMITTED	----NO---	000007000362
*RELAY	STATES-IN-UN/OWNS/CITY-DATA	----NO---	000006000464

***** END RELAYS *****

Figure 5-12 (cont'd)
Physical Dump SDDL Tables

IDS ANALYZER REPORT

IDS DATABASE = OLD-PRESIDENTIAL

	IDS-NAME	ADBMS NAME	DISPL. (CHAR)	STORAGE CONSTRUCT	PAD CHAR	IDS/RIF LENGTH	VALUE
GROUP-	ADMINISTRATION	ADMINI	0				3
WITH ITEMS-							
KEY	ADMIN-NUM	ADMIN-	9	ALPHANUMERIC	" "	3/ 3	
	MONTHC	MONTHC	12	ALPHANUMERIC	" "	10/ 10	
	DAYEC	DAYEC	22	ALPHANUMERIC	" "	2/ 2	
	YEARC	YEARC	24	ALPHANUMERIC	" "	4/ 4	
	NAME-FIRSTN	NAME-F	28	ALPHANUMERIC	" "	10/ 10	
	NAME-LASTN	NAME-L	38	ALPHANUMERIC	" "	10/ 10	
	ST-NAME	ST-NAM	87	ALPHANUMERIC	" "	60/ 60	
ET-SIG*	FIRSTN<P-A>	FIRST>					
ET-SIG*	INIT<P-A>	INIT<P					
ET-SIG*	LASTN<P-A>	LASTN<					
MASTER OF-							5
	ADMINISTRATI/OWNS/STATE-DREF	ADMINF		CONCAT			13
	STATES-ADMITTED	STATED		CHAIN			
DETAIL OF-							
	CALC-ADMINISTRATION	CALC-A		CHAIN			
	P-A	P-A		CHAIN			
GROUP-	STATE-DREF	STATE-	51				
WITH ITEMS-							
	STATE-DREF/IT	STATE-	0	INTEGER-FW	"0"	6/ 6	
S-S-KEY*	ADMIN-NUM<ADMINISTRATI/OWNS/STATE-DR EF>	ADMIN>					
SET-SIG*	STATE-NAME<ADMITTED-BY-ADMIN>	STATE>					
MASTER OF-							
DETAIL OF-							
	ADMINISTRATI/OWNS/STATE-DREF	ADMINF		CONCAT			
	ADMITTED-BY-ADMIN	ADMITT		CHAIN			

Figure 5-13
User dump SDDL tables

IDS ANALYZER REPORT

RELATIONS

	IDS-NAME	ADBMS NAME	RELATION TYPE
RELATION	ADMINISTRATI/OWNS/STATE-DREF	ADMINF	CONCAT
MASTER	ADMINISTRATION	ADMINI	
DETAIL	STATE-DREF	STATE-	
RELATION	ADMITTED-BY-ADMIN	ADMITT	CHAIN
MASTER	STATES-IN-UNION	STATES	
DETAIL	STATE-DREF	STATE-	
RELATION	CALC-ADMINISTRATION	CALC-A	CHAIN
MASTER	SYSTEM		
DETAIL	ADMINISTRATION	ADMINI	
RELATION	CALC-CONGRESS	CALC-C	CHAIN
MASTER	SYSTEM		
DETAIL	CONGRESS	CONGRE	
RELATION	CALC-ELECTION	CALC-E	CHAIN
MASTER	SYSTEM		
DETAIL	ELECTION	ELECTI	
RELATION	CALC-PRESIDENT	CALC-P	CHAIN
MASTER	SYSTEM		
DETAIL	PRESIDENT	PRESID	

SYSTEM ENTRY RELATIONS		
CALC-ADMINISTRATION	CALC-A	CHAIN
CALC-CONGRESS	CALC-C	CHAIN
CALC-ELECTION	CALC-E	CHAIN
CALC-PRESIDENT	CALC-P	CHAIN
CALC-STATES-IN-UNION	CALC-S	CHAIN

Figure 5-13 (cont'd)
User dump SDDL tables

IDS DATABASE	The name of the database for which the output below is collected. Each database defined within the combined extended MD section is printed. Note that this name is the same as that supplied on the run-time parameter file.
IDS NAME	User names supplied in the extended MD section. The user refers to these names in writing the TDL.
ADBMS NAME	Every user name has a corresponding six character ADBMS name. In some Translator error messages, the ADBMS name is output, hence, this listing is a useful cross-reference tool.
DISPL.(CHAR)	Displacement in characters of the group (if a contained-in-repeating group) or item. Note in the example that ADMIN-NUM starts at location ten (10) since there are nine characters taken up by the IDS record header.
STORAGE CONSTRUCT	For items, the storage representation is printed; for relations, the relation type is denoted. Possible relation types are: CONCAT - concatenated relations CHAIN - phantom pointers or IDS chains MTCHKY - Match-key relations Note that items that are PIC 99 are printed as *alphanumeric-type*.
PAD-CHAR	By default, alphanumeric items have a pad character of blank; computational items have a pad character of zero. This can be overridden by use of the "61 PAD" entry, however.
IDS/RIF LENGTH	The first number to the left of the slash(/) is the length in characters of the item in the IDS database. This should agree with the user's view of the database. The number to the right of the slash is the length in characters of the item as it is represented in the RIF. Note that all integers are single precision in the RIF and all floating point items are double precision in the RIF.
VALUE	For IDS records, this is the number specified by the TYPE IS clause. For ISP and sequential database records, the character string specified by the 61 IDENT entry is printed. Note that contained-in-repeating groups do not have a value. The Reader identifies contained-in-repeating groups by their displacement within their "containing" groups.
KEY	To the left of any item which was declared to be a primary key is the indication of *KEY*.
S-S-KEY	Any set-significant items defined in the EXTERNAL-KEYS-FROM level 61 will be denoted as a set-significant-key (S-S-KEY).

SET-SIG	All set-significant items for the group are identified by *SET-SIG* if they are not primary keys. In the example, because ADMINISTRATION is a detail of the relation P-A (whose master has three primary keys; LASTN, FIRSTN, INIT), three set-significant items must be present within ADMINISTRATION. Note that they are not physically present, only logically. Their names are constructed according to the rules given in Section 3.2.3 of the User Manual.
MASTER OF	Each relation of which the group is a master is listed along with the corresponding ADBMS name and relation type. Note in the example the construction of the concatenated relation name from ADMINISTRATION to STATE-DREF.
DETAIL OF	Each relation of which the group is a detail is listed along with the corresponding ADBMS name and relation type. Note in the example that since ADMINISTRATION is a CALC record, it must be a detail of a relation (whose owner is SYSTEM) named CALC-ADMINISTRATION.

After all the groups within each database have been printed, the last few pages of the report give a summary of all the relations (in alphabetical order) represented within the SDDL tables in question. Lastly, all system entry relations are listed for useful reference.

6.0 RUNNING THE TDL ANALYZER

After the user has created both the source and the target Stored Data Definition Language (SDDL) tables, it is necessary to supply the Translator with a description of the mapping (restructuring) to be performed by the Restructurer. This is done by supplying the Translation Definition Language (TDL) Analyzer with a textual description of the mapping. The TDL Analyzer then produces tables which are used by the Restructurer to perform the mapping. The general processing flow of the TDL Analyzer is as follows:

1. Make temporary copies of the source and target SDDL tables.
2. Analyze the TDL description and produce the TDL tables.
3. Post-process the tables by including information which improves the run-time performance of the Restructurer.
4. Optionally give a user-friendly dump of the TDL tables.

Before executing the TDL Analyzer, the following steps must be performed:

1. The IDS Analyzer must be run to produce source and target SDDL tables (see Section 5.0 for a description of this process).
2. A valid TDL description must be written (see Section 4.0 for a description of this process).
3. A permanent file (random) must be created to hold the TDL table database. A rule of thumb to use for determining the size of this file is to use 18 llinks or one-half the size of the target SDDL database, whichever is greater. This formula will usually supply a TDL table size which is slightly larger than needed.

6.1 Detailed Overview of TDL Analyzer Execution

Executing the TDL Analyzer requires the five activities shown below:

1. Make temporary copies of the source and target SDDL tables.
2. Execute the TDL Analyzer to produce the TDL tables.
3. Post-process the TDL tables with the Sysacc builder.
4. Post-process the TDL tables with the Compatibility builder.
5. Optionally give a user-friendly dump of the TDL tables.

The entire process is illustrated in Figure 6-1; the components are discussed below.

UTILITY — The GCOS utility program. See the UTILITY manual for a description of the output and error reports generated by this module.

SOURCE and TARGET SDDL TABLES — ADBMS databases produced by the IDS Analyzer which contain the Data Translator's descriptions of the source and target databases.

SCRATCH COPY SOURCE and TARGET SDDL TABLES — Temporary files which contain copies of the source and target SDDL tables.

UTILITY REPORT — A report on the success or failure of the utility program.

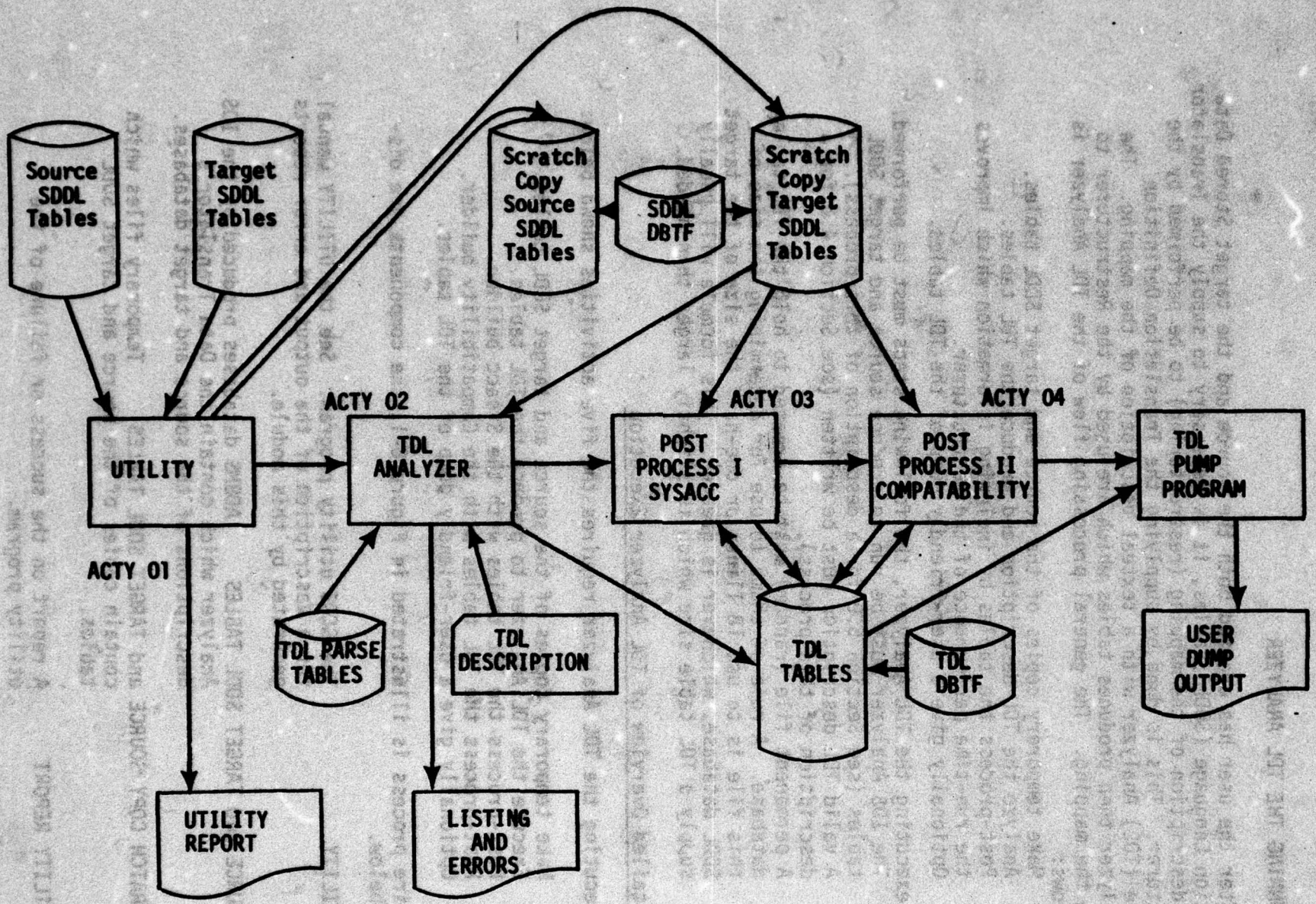


Figure 6-1
TDL Analyzer Components

TDL ANALYZER	An R* file containing the object version of the TDL Analyzer.
TDL PARSE TABLES	A file which contains the TDL parsing tables.
TDL description	The user-written TDL description.
SDDL DBTF	The database tables file describing to ADBMS the format and contents of the SDDL tables databases. See Appendix F.
TDL Tables	An ADBMS database containing the Data Translator representation of the restructuring mapping.
TDL DBTF	The database tables file describing to ADBMS the format and contents of the TDL tables database. See Appendix F.
LISTING AND ERRORS	A listing (if requested) of the TDL description, user errors and warnings, and a statistical summary of the TDL Analyzer execution.
POST PROCESS I - SYSACC	An R* file containing the object version of the first post-process.
POST PROCESS II - Compatibility	An R* file containing the object version of the second post-process.
TDL Dump Program	An R* file containing the object version of the program which produces the user dump of the TDL tables.
USER DUMP OUTPUT	The user dump of the TDL tables.

6.2 Explanation of Processing Flows

This section comprises a brief overview of the TDL Analyzer. Each activity is described in turn with a complete JCL description given in later sections.

The TDL Analyzer operates by sequentially reading each card image in the TDL description. Each card image is the checked-for syntax and semantic errors. If no errors are found, the SDDL tables are accessed and entries are made in the TDL tables (when appropriate).

6.3 TDL Analyzer JCL

Control cards for executing all five TDL Analyzer activities are illustrated in Figure 6-2. This sequence of control cards is available on the SYSGEN tape containing the Data Translator. Each activity is described below.

```
1010  $    IDENT
1020  $    NOTE   ***********************************************
1030  $    NOTE   *                                             *
1040  $    NOTE   *     COPY SOURCE AND TARGET SDDL TABLES      *
1050  $    NOTE   *                                             *
1060  $    NOTE   ***********************************************
1070  $    UTILITY
1080  $    FUTIL  14,15,RWD/14/,RCOPY/1F/
1090  $    PRMFL  14,R,R,SOURCE SDDL DATA BASE FILE
1100  $    FILE   15,X1S,SIZE OF SOURCE SDDL DATA BASE FILE IN LINKS
1110  $    FUTIL  16,17,RWD/16/,RCOPY/1F/
1120  $    PRMFL  16,R,R,TARGET SDDL DATA BASE FILE
1130  $    FILE   17,X2S,SIZE OF TARGET SDDL DATA BASE FILE IN LINKS
1140  $    NOTE   ***********************************************
1150  $    NOTE   *                                             *
1160  $    NOTE   *          EXECUTE TDL ANALYZER               *
1170  $    NOTE   *                                             *
1180  $    NOTE   ***********************************************
1190  $    EXECUTE NREST
1200  $    LIMITS XX,46K,-3K,XX   USE APPROPRIATE LIMITS
1210  $    PRMFL  R*,R,S,TDL RSTAR FILE
1220  $    FILE   H*,Z1R,40R
1230  $    PRMFL  04,R,S,TDL PARSE TABLES
1240  $    DATA   02,,COPY
1250  $    SELECTA TDL DESCRIPTION
1260  $    ENDCOPY
1270  $    NOTE   ANALYZER OUTPUT IS ON FILE CODE 06
1280  $    FILE   07,X1R
1290  $    FILE   09,X2S
1300  $    PRMFL  11,W,R,TDL DATA BASE FILE
1310  $    PRMFL  12,R,S,ADBMS TABLE FILE FOR TDL TABLES
1320  $    FILE   13,NULL
1330  $    FILE   14,NULL
1340  $    PRMFL  TR,R,R,TRANSLATOR LIBRARY
1350  $    IF     18,IF          IF TDL ERRORS THEN SKIP TO DUMP
1360  $    NOTE   ***********************************************
1370  $    NOTE   *                                             *
1380  $    NOTE   *     POST PROCESS I - SYSACC BUILDER         *
1390  $    NOTE   *                                             *
1400  $    NOTE   ***********************************************
1410  $    EXECUTE NREST
1420  $    LIMITS XX,33K,-3K,XX   USE APPROPRIATE LIMITS
1430  $    PRMFL  R*,R,S,SYSACC BUILDER RSTAR FILE
1440  $    PRMFL  07,W,R,TDL DATA BASE FILE
```

Figure 6-2

TDL Analyzer Prototype JCL

```
1440    $       PRMFL   07,W,R,TDL DATA BASE FILE
1450    $       FILE    08,X2S
1460    $       PRMFL   TR,R,R,TRANSLATOR LIBRARY
1470    $       NOTE    **********************************************
1480    $       NOTE    *                                            *
1490    $       NOTE    *  POST PROCESS II - COMPATIBILITY BUILDER   *
1500    $       NOTE    *                                            *
1510    $       NOTE    **********************************************
1520    $       EXECUTE NREST
1530    $       LIMITS  XX,52K,-3K,XX   USE APPROPRIATE LIMITS
1540    $       PRMFL   R*,R,S,COMPATIBILITY RSTAR FILE
1550    $       PRMFL   07,W,R,TDL DATA BASE FILE
1560    $       FILE    08,X2S
1570    $       PRMFL   TR,R,R,TRANSLATOR LIBRARY
1580    $       NOTE    **********************************************
1590    $       NOTE    *                                            *
1600    $       NOTE    *              TDL TABLE DUMPER              *
1610    $       NOTE    *                                            *
1620    $       NOTE    **********************************************
1630    $       IF      19,ENDJOB       SKIP DUMP IF NOT REQUESTED
1640    $       EXECUTE NREST
1650    $       LIMITS  XX,25K,-3K,XX   USE APPROPRIATE LIMITS
1660    $       PRMFL   R*,R,S,TDL DUMPER RSTAR FILE
1670    $       PRMFL   02,R,R,TDL DATA BASE FILE
1680    $       NOTE    DUMP OUTPUT IS ON FILE CODE 06
1690    $       PRMFL   TR,R,R,TRANSLATOR LIBRARY
1700    $       ENDJOB
```

Figure 6-2 (cont.)

6.4 TDL Analyzer - Activity 1

```
1020    $    NOTE    ************************************************
1030    $    NOTE    *                                              *
1040    $    NOTE    *     COPY SOURCE AND TARGET SDDL TABLES       *
1050    $    NOTE    *                                              *
1060    $    NOTE    ************************************************
1070    $    UTILITY
1080    $    FUTIL   14,15,RWD/14/,RCOPY/1F/
1090    $    PRMFL   14,R,R,SOURCE SDDL DATA BASE FILE
1100    $    FILE    15,X1S,SIZE OF SOURCE SDDL DATA BASE FILE IN LINKS
1110    $    FUTIL   16,17,RWD/16/,RCOPY/1F/
1120    $    PRMFL   16,R,R,TARGET SDDL DATA BASE FILE
1130    $    FILE    17,X2S,SIZE OF TARGET SDDL DATA BASE FILE IN LINKS
```

Filecode	LUD	Description
14	-	Source SDDL tables
16	-	Target SDDL tables
18	X1S	Scratch copy of source SDDL tables
17	X2S	Scratch copy of target SDDL tables

Example of Output

Figure 6-3 is a sample of a successful execution of activity 1. The report is generated by the UTILITY program on report code 53. If the UTILITY operation is not successful, then a message indicating the problem will be printed. These messages are described in the UTILITY manual.

Discussion

Activity one makes temporary copies of the source and target SDDL tables. The user should enter the file names of the source and target SDDL tables where indicated on lines 1090 and 1120 respectively. The sizes of these files, rounded up to the closest link (12 llinks) should be entered where indicated on lines 1000 and 1130.

```
2278T 01  12/20/76          UTILITY  REPORT  750204          PAGE     1

$       FUTIL   14.15.RWD/14/.RCOPY/1F/
 RCOPYD              1 FILE

$       FUTIL   16.17.RWD/16/.RCOPY/1F/
 RCOPYD              1 FILE
```

Figure 6-3

Activity 1 Example Output

6.5 TDL Analyzer - Activity 2

```
1140  $   NOTE     **********************************************
1150  $   NOTE     *                                            *
1160  $   NOTE     *              EXECUTE TDL ANALYZER          *
1170  $   NOTE     *                                            *
1180  $   NOTE     **********************************************
1190  $   EXECUTE  NREST
1200  $   LIMITS   XX,46K,-3K,XX   USE APPROPRIATE LIMITS
1210  $   PRMFL    R*,R,S,TDL RSTAR FILE
1220  $   FILE     H*,Z1R,40R
1230  $   PRMFL    04,R,S,TDL PARSE TABLES
1240  $   DATA     02,,COPY
1250  $   SELECTA  TDL DESCRIPTION
1260  $   ENDCOPY
1270  $   NOTE     ANALYZER OUTPUT IS ON FILE CODE 06
1280  $   FILE     07,X1R
1290  $   FILE     09,X2S
1300  $   PRMFL    11,W,R,TDL DATA BASE FILE
1310  $   PRMFL    12,R,S,ADBMS TABLE FILE FOR TDL TABLES
1320  $   FILE     13,NULL
1330  $   FILE     14,NULL
1340  $   PRMFL    TR,R,R,TRANSLATOR LIBRARY
1350  $   IF       18,IF              IF TDL ERRORS THEN SKIP TO DUMP
```

Filcode	LUD	Description
M*	Z1R	A required filecode so an entry can be made in PAT.
R*	---	TDL Analyzer R* file from the SYSGEN tape.
04	---	The TDL parsing tables from the SYSGEN tape.
02	---	The TDL description (in BCD).
06	---	TDL Analyzer output.
07	X1R	A scratch copy of the SOURCE SDDL tables.
09	X2S	A scratch copy of the target SDDL tables.
13, 14	---	Null files (for dump I/O).
TR	---	Random library of Data Translator object routines from the SYSGEN tape.

Example of Output

Figure 6-4 shows a sample of the TDL Analyzer output. TDL Analyzer error messages are printed on I/O unit 06. Due to the parsing algorithm, most error messages appear two lines below the actual error. Thus, the user should examine the two lines which precede an error message for the occurrence of the error. The error messages are discussed in Appendix B. Certain errors cause the TDL Analyzer to enter error recovery mode. When this occurs, the TDL Analyzer is attempting to continue processing by skipping any input lines that have been rendered invalid due to a previous error. The TDL Analyzer prints error messages indicating the points where the error recovery procedure was invoked and terminated. Any intervening input lines were not processed in any way.

Discussion

Activity two executes the TDL Analyzer. The user should place appropriate file names in the JCL as indicated. The user should use limits appropriate for the given job in line 1200. Most TDL Analyzer executions require less than 0.1 hours of CPU time.

6.6 TDL Analyzer - Activity 3

```
1360    $       NOTE    ********************************************
1370    $       NOTE    *                                          *
1380    $       NOTE    *      POST PROCESS I - SYSACC BUILDER     *
1390    $       NOTE    *                                          *
1400    $       NOTE    ********************************************
1410    $       EXECUTE NREST
1420    $       LIMITS  XX,33K,-3K,XX   USE APPROPRIATE LIMITS
1430    $       PRMFL   R*,R,S,SYSACC BUILDER RSTAR FILE
1440    $       PRMFL   07,W,R,TDL DATA BASE FILE
1450    $       FILE    08,X2S
1460    $       PRMFL   TR,R,R,TRANSLATOR LIBRARY
```

Filecode	LUD	Description
R*	---	Post-process J R* from SYSGEN tape.
07	---	TDL tables.
08	X2S	Scratch copy of target SDDL tables.
TR	---	Translator library from SYSGEN tape.

```
UNIVERSITY OF MICHIGAN ANALYZER(VER. 2-2) 11.41.08    DEC 20. 1976      PAGE   1

DATA BASE INITIALIZED WITH      7 PAGES.
    1 >/*   NEW PREZ TO PREZ PARTS STATES  */
    2 >/*  $D  */
    3 >TARGET RECORD CITIES
    4 >    TDLAP CITI
    5 >        SOURCE RECORD STATES-IN-UNION ACCESS VIA CALC-STATES-IN-UNION
    6 >            SOURCE RECORD CITIES ACCESS VIA HAS-LARGE-CITIES
    7 >                ACTUAL DATA IN ORDER
    8 >                SET SIGNIFICANT DATA BY NAME
    9 >TARGET RECORD OCCUP-MARR
   10 >    TDLAP OCCU
   11 >        SOURCE RECORD PRESIDENT ACCESS VIA CALC-PRESIDENT
   12 >            FIRST-NAME ASSIGN TO PRES-S-FNAME<PRES-INFO>
   13 >            INIT ASSIGN TO PRES-S-MNAME<PRES-INFO>
   14 >            LAST-NAME ASSIGN TO PRES-S-LNAME<PRES-INFO>
   15 >                SOURCE RECORD OCCUPATION ACCESS VIA HAD-OCCUPATION
   16 >                    TITLE-OF-JOB ASSIGN TO TITLE-OF-JOB
   17 >                SOURCE RECORD MARRIAGE-DATA ACCESS VIA
   18 >                        MARRIAGE-INFO FROM PRESIDENT
   19 >                    WIVES-NAME ASSIGN TO WIVES-NAME
   20 >                    MONTH-MARRIED ASSIGN TO MONTH-MARRIED
   21 >                    DAY-MARRIED ASSIGN TO DAY-MARRIED
   22 >                    YEAR-MARRIED ASSIGN TO YEAR-MARRIED
   23 >                    NUMBER-OF-CHILDREN ASSIGN TO NUMBER-OF-CHILDREN
   24 >TARGET RECORD PRESIDENT-S
   25 >    TDLAP PRES
   26 >        SOURCE RECORD PRESIDENT ACCESS VIA CALC-PRESIDENT
   27 >            SET SIGNIFICANT DATA BY NAME
   28 >            FIRST-NAME ASSIGN TO PRES-S-FNAME
   29 >            INIT ASSIGN TO PRES-S-MNAME
   30 >            LAST-NAME ASSIGN TO PRES-S-LNAME
   31 >TARGET RECORD STATES-S
   32 >    TDLAP STAT
   33 >        SOURCE RECORD STATES-IN-UNION ACCESS VIA CALC-STATES-IN-UNION
   34 >            STATE-NAME ASSIGN TO STATE-NAME
   35 >            YEAR-ADMITTED ASSIGN TO YEAR-ADMITTED
   36 >            CAPITAL ASSIGN TO CAPITAL
   37 >            AREA-SQ-MI ASSIGN TO AREA-SQ-MI
   38 >            AREA-RANK ASSIGN TO AREA-RANK
   39 >            POPULATION ASSIGN TO POPULATION
   40 >            POP-RANK ASSIGN TO POP-RANK
   41 >            ELECTORAL-VOTES ASSIGN TO ELECTORAL-VOTES
   42 >        SOURCE RECORD STATES-ADMITTED ACCESS VIA ADMITTED-DURING
   43 >            ACCEPT IF NULL
   44 >                SOURCE RECORD ADMINISTRATION ACCESS VIA
   45 >                        ADMITTED-STATES
   46 >                    ACCEPT IF NULL
   47 >                ADMINISTRATION-NUMBER ASSIGN TO
   48 >                        ADMITTED-BY-ADMIN
   49 >                    NULL VALUE = '   '
   50 > EOF
```

Figure 6-4

Activity 2 Example Output

```
UNIVERSITY OF MICHIGAN ANALYZER(VER. 2-2) 11.42.36     DEC 20, 1976     PAGE   2

        ANALYZER STATISTICS

INITIALIZATION TIME =     0.65 SECONDS
PROCESSING TIME     =    11.43 SECONDS
DUMPING TIME        =     0.04 SECONDS
TOTAL TIME          =    12.11 SECONDS
  50 CARDS WERE PROCESSED
PROCESSING RATE=  262 CARDS/MINUTE
  0 ERROR(S) AND   0 WARNING(S) WERE DETECTED
```

Figure 6-4 (cont.)

Example of Output

Figure 6-5 shows the output generated by this activity.

Discussion

The user should supply appropriate file names and limits as indicated. Most executions of this activity require less than 0.1 hours of CPU time.

6.7 TDL Analyzer - Activity 4

```
1470    $       NOTE    ***********************************************
1480    $       NOTE    *                                             *
1490    $       NOTE    *  POST PROCESS II - COMPATIBILITY BUILDER    *
1500    $       NOTE    *                                             *
1510    $       NOTE    ***********************************************
1520    $       EXECUTE NREST
1530    $       LIMITS  XX,52K,-3K,XX   USE APPROPRIATE LIMITS
1540    $       PRMFL   R*,R,S,COMPATIBILITY RSTAR FILE
1550    $       PRMFL   07,W,R,TDL DATA BASE FILE
1560    $       FILE    08,X2S
1570    $       PRMFL   TR,R,R,TRANSLATOR LIBRARY
```

Filecode	LUD	Description
R*	---	Post-process II R* from SYSGEN tape.
07	---	TDL table.
08	X2S	Scratch copy of target SDDL tables.
TR	---	Translator library from SYSGEN tape.

Example of Output

Figure 6-6 shows the output generated by this activity.

Discussion

The user should supply appropriate file names and limits as indicated. Most executions of this activity require less than 0.1 hours of CPU time.

```
*****SYSACCS SUCCESSFULLY COMPUTED AND SET*****
 LINE # = 25, IERR =  0
```

Figure 6-5

Activity 3 Example Output

```
#####COMPATIBILITY SETS CONSTRUCTED#####
```

Figure 6-6

Activity 4 Example Output

6.8 TDL Analyzer - Activity 5

```
1580    $    NOTE    **********************************************
1590    $    NOTE    *                                            *
1600    $    NOTE    *              TDL TABLE DUMPER              *
1610    $    NOTE    *                                            *
1620    $    NOTE    **********************************************
1630    $    IF      19,ENDJOB          SKIP DUMP IF NOT REQUESTED
1640    $    EXECUTE NREST
1650    $    LIMITS  XX,25K,-3K,XX      USE APPROPRIATE LIMITS
1660    $    PRMFL   R*,R,S,TDL DUMPER RSTAR FILE
1670    $    PRMFL   02,R,R,TDL DATA BASE FILE
1680    $    NOTE    DUMP OUTPUT IS ON FILE CODE 06
1690    $    PRMFL   TR,R,R,TRANSLATOR LIBRARY
```

Filecode	LUD	Description
R*	---	R* file for TDL table. Dumper from SYSGEN tape.
02	---	TDL tables.
06	---	TDL dump output.
TR	---	Translator library from SYSGEN tape.

Example of Output

Figure 6-5 shows an example of part of the TDL dump output. The TDL user dump output provides the user with a description of the TDL tables that were produced. The dump is arranged by target records. It includes many items of information with which the inexperienced or casual user should not be concerned. The TDL user dump is primarily provided for those users whose restructuring needs call for unorthodox or complicated TDL descriptions. The casual user may, however, find it a useful device for verifying that the TDL description is correct.

Discussion

The user should supply the appropriate file names and limits as indicated. Most executions of this activity require less than 0.1 hours of CPU time.

```
TARGET_RECORD PRESID, SYSSET SYS112, ACCESS_PATH   1, KEY=000002001130.

   ACCESS_PATH PRES          COMPATIBLE_ABOVE
                                OCCU

      SOURCE_RECORD PRESID HAS_KEY=000002001204.  ACCESS_IS_VIA CALC-P, OWNED_BY  PRES        , A_ROOT_ALONG_OUR
      WITH_KEY=000002001146.  REJECT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE RIELEN RIEPAD TARGET_ITEM_ASSIGNED
         FIRST-      000002001340 000000                            0     10        PRES-E
         INIT        000002001404 000000                            0     1         PRES-M
         LAST-N      000002001450 000000                            0     10        PRES-S

      SOURCE_RECORD STATES HAS_KEY=000002001240.  ACCESS_IS_VIA HAS-NA, OWNED_BY  PRESID     , A_NODE_ALONG_NAO
      WITH_KEY=000002001204.  ACCEPT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE RIELEN RIEPAD TARGET_ITEM_ASSIGNED
         STATE-      000002001274 000000                            0     10        STATE>


TARGET_RECORD OCCUP-, SYSSET SYS11=, ACCESS_PATH   2, KEY=000002000224.

   ACCESS_PATH OCCU                          COMPATIBLE_BELOW
                                                PRES

      SOURCE_RECORD PRESID HAS_KEY=000002000300.  ACCESS_IS_VIA CALC-P, OWNED_BY  OCCU        , A_ROOT_ALONG_OUR
      WITH_KEY=000002000242.  REJECT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE RIELEN RIEPAD TARGET_ITEM_ASSIGNED
         FIRST-      000002000334 000000                            0     10        PRES-S
         INIT        000002000400 000000                            0     1         PRES-O
         LAST-N      000002000444 000000                            0     10        PRES->

      SOURCE_RECORD OCCUPA HAS_KEY=000002000510.  ACCESS_IS_VIA HAD-OC, OWNED_BY  PRESID     , A_NODE_ALONG_OUR
      WITH_KEY=000002000300.  REJECT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE RIELEN RIEPAD TARGET_ITEM_ASSIGNED
         TITLE-      000002000544 000000                            0     10        TITLE-

      SOURCE_RECORD MARRIT HAS_KEY=000002000610.  ACCESS_IS_VIA MARRIA, OWNED_BY  PRESID     , A_NODE_ALONG_OUR
      WITH_KEY=000002000300.  REJECT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE RIELEN RIEPAD TARGET_ITEM_ASSIGNED
         DAY-MA      000002000754 000000                            0     10        DAY-MA
         MONTH-      000002000710 000000                            0     10        MONTH-
         NUMBER      000002001064 000000                            0     10        NUMBER
         WIVES-      000002000644 000000                            0     10        WIVES-
         YEAR-M      000002001020 000000                            0     10        YEAR-M
```

```
TARGET RECORD CITIES, SYSSET SYS11*, ACCESS_PATH   1, KEY=000002000014.

   ACCESS_PATH CITI                          COMPATIBLE_BELOW
                                                STAT

      SOURCE_RECORD STATES HAS_KEY=000002000070.  ACCESS_IS_VIA CALC-S, OWNED_BY  CITI        , A_ROOT_ALONG_OUR
      WITH_KEY=000002000032.  REJECT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE FIELEN FIEPAD TARGET_ITEM_ASSIGNED
         STATE-       000002000160 000000                          0      10        STATE>

      SOURCE_RECORD CITIES HAS_KEY=000002000124.  ACCESS_IS_VIA HAS-LA, OWNED_BY  STATES      , A_NODE_ALONG_OUR
      WITH_KEY=000002000070.  REJECT_IF_NULL.  ACTUAL_DATA_BASE=_BASE= 16.



TARGET RECORD STATES, SYSSET SYS111, ACCESS_PATH   1, KEY=000002001514.

   ACCESS_PATH STAT          COMPATIBLE_ABOVE
                                CITI

      SOURCE_RECORD STATES HAS_KEY=000002001570.  ACCESS_IS_VIA CALC-S, OWNED_BY  STAT        , A_ROOT_ALONG_OUR
      WITH_KEY=000002001532.  REJECT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE FIELEN FIEPAD TARGET_ITEM_ASSIGNED
         AREA-R       000003000047 000000                          0       6 0      AREA-R
         AREA-S       000003000003 000000                          0       6 0      AREA-S
         CAPITA       000002001734 000000                          0      10        CAPITA
         ELECTO       000003000223 000000                          0       6 0      ELECTO
         POP-RA       000003000157 000000                          0       6 0      POP-RA
         POPULA       000003000113 000000                          0       6 0      POPULA
         STATE-       000002001624 000000                          0      10        STATE-
         YEAR-A       000002001670 000000                          0       4        YEAR-A

      SOURCE_RECORD STATED HAS_KEY=000003000267.  ACCESS_IS_VIA ADMITT, OWNED_BY  STATES      , A_NODE_ALONG_OUR
      WITH_KEY=000002001570.  ACCEPT_IF_NULL.

      SOURCE_RECORD ADMINI HAS_KEY=000003000323.  ACCESS_IS_VIA ADMITS, OWNED_BY  STATED      , A_NODE_ALONG_NXO
      WITH_KEY=000003000267.  ACCEPT_IF_NULL.

         SOURCE_ITEM - KEY_________ CONVERSION_ROUTINE CONVERSION_TYPE FIELEN FIEPAD TARGET_ITEM_ASSIGNED
         ADMINI       000003000357 000000                          0       3        ADMITT
            THE NULL VALUE=//
```

7.0 THE READER MODULE

The fourth step in the Data Translation process is running the Reader. The Reader is the first module to work with the user's data. It converts the user's WWDMS sequential, ISP or IDS database to one ADBMS database called the source RIF. The source RIF (SRIF) is logically equivalent to the source database(s). The Reader is driven by information stored in the source SDDL tables. Before the Reader can be run the following steps must be completed:

1. The source SDDL tables must be successfully created.
2. The source database(s) should be available. MD sections for all IDS databases should be available.
3. All necessary files (see Section 7.4) should be created.

7.1 Reader Submodules

The Version IIA Release 2 Data Translator has three Reader modules: one each for WWDMS sequential, ISP and IDS databases. Each Reader performs the same function, that is, produces the source RIF, but due to the differences of each of the file systems the Readers consist of different submodules to improve operating efficiency. The architecture of each Reader is shown in Figure 7-1, 7-2, and 7-3. Figure 7-4 shows the Reader configuration for one IDS, one ISP and two WWDMS sequential databases. Although Figure 7-4 is probably not a typical example, it shows the use of all three Readers to create one SRIF.

The Reader submodules of which the user should be aware and should understand are the following:

1. Accessor (IDS, ISP, sequential) - the Accessor retrieves logical records from the database. The IDS Accessor is a COBOL-IDS program which only reads pages and returns information describing each logical record. Note that the IDS Accessor must be compiled with the MD section for the database before the IDS Reader can be run.
2. Populator (Network, Hierarchical) - the Populator controls the linking of sets in the SRIF. It duplicates the logical relationships that exist in the source database(s) in the SRIF. The network Populator uses an internal database called the Deferred Reference Table (DRT). The DRT contains information about sets in the SRIF.
3. Reader Control - the Reader Control submodule is essentially the same for each Reader. It performs initialization and wrapup steps, moves data to the SRIF, links SYSTEM-owned sets, and links match-key relations.

7.2 Reader Processing Flow

Each Reader uses the same type of algorithm, only capabilities differ. The first step of the Reader algorithm is initialization. During initialization the DDL Writer module is executed. It writes the ADBMS DDL for the SRIF using information in the source SDDL tables. Then, if this is the first Reader run, the SRIF is initialized and (if IDS Reader) the DRT database is initialized.

AD-A046 806 MICHIGAN UNIV ANN ARBOR GRADUATE SCHOOL OF BUSINESS--ETC F/G 9/2
DATA TRANSLATOR VERSION IIA, RELEASE 2 USER MANUAL REVISION 1.(U)
MAR 77 E KINTZER, J BODWIN, W COOL DCA100-75-C-0064
UNCLASSIFIED WP-DT-3.4 CCTC-CSM-UM-236-77-REV-1 NL

3 OF 4
AD A046806

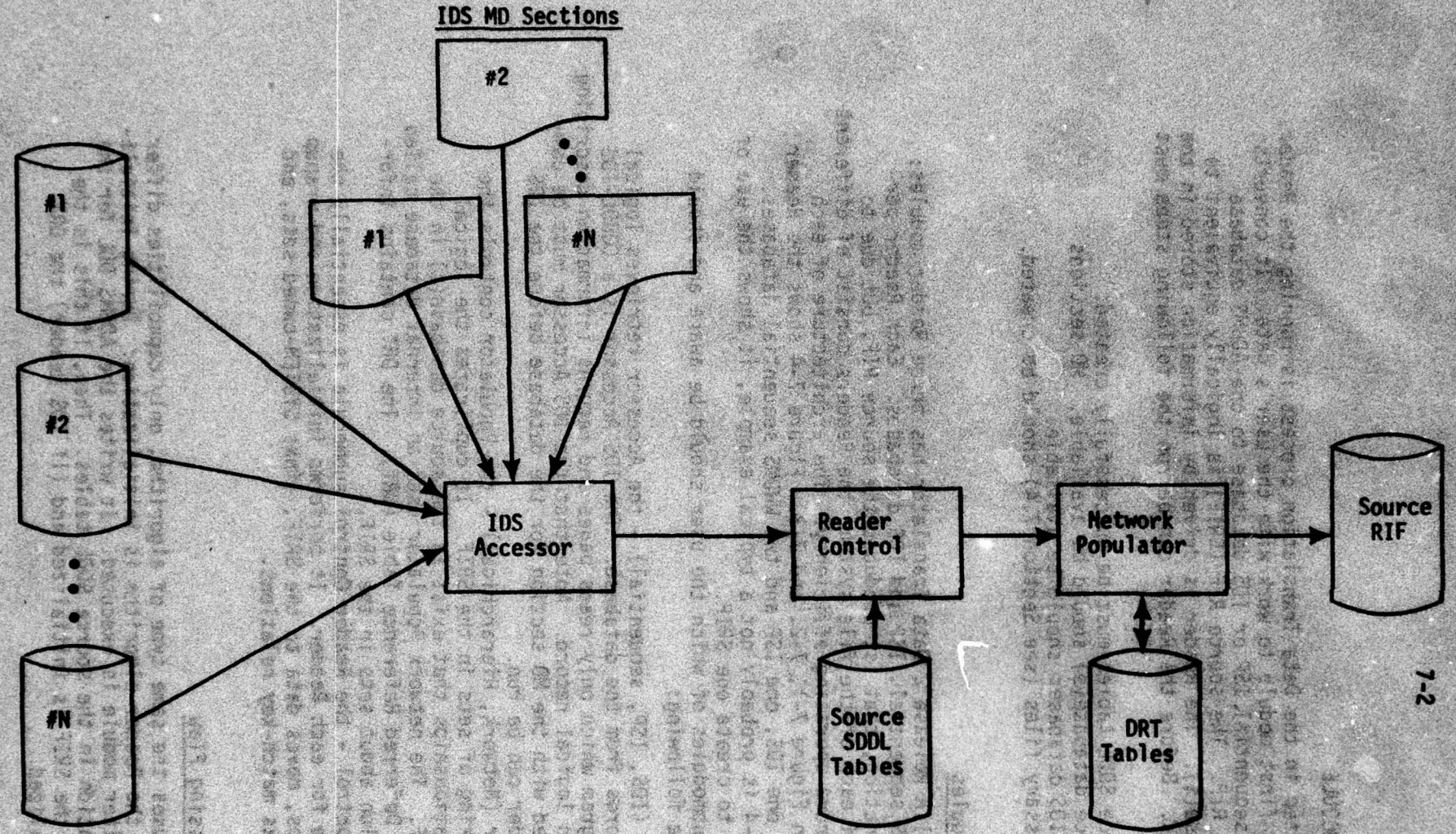


Figure 7-1
IDS Reader

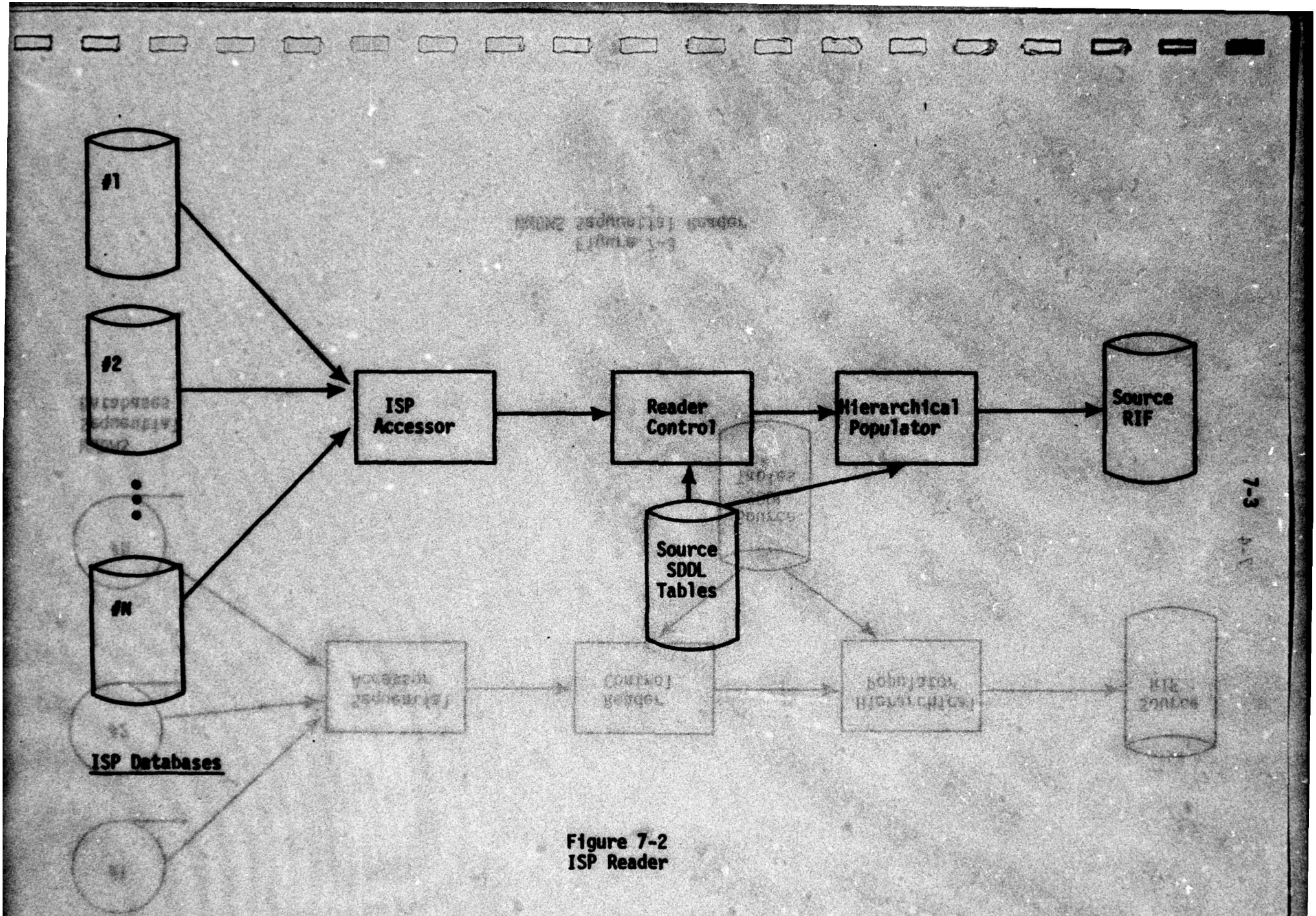


Figure 7-2
ISP Reader

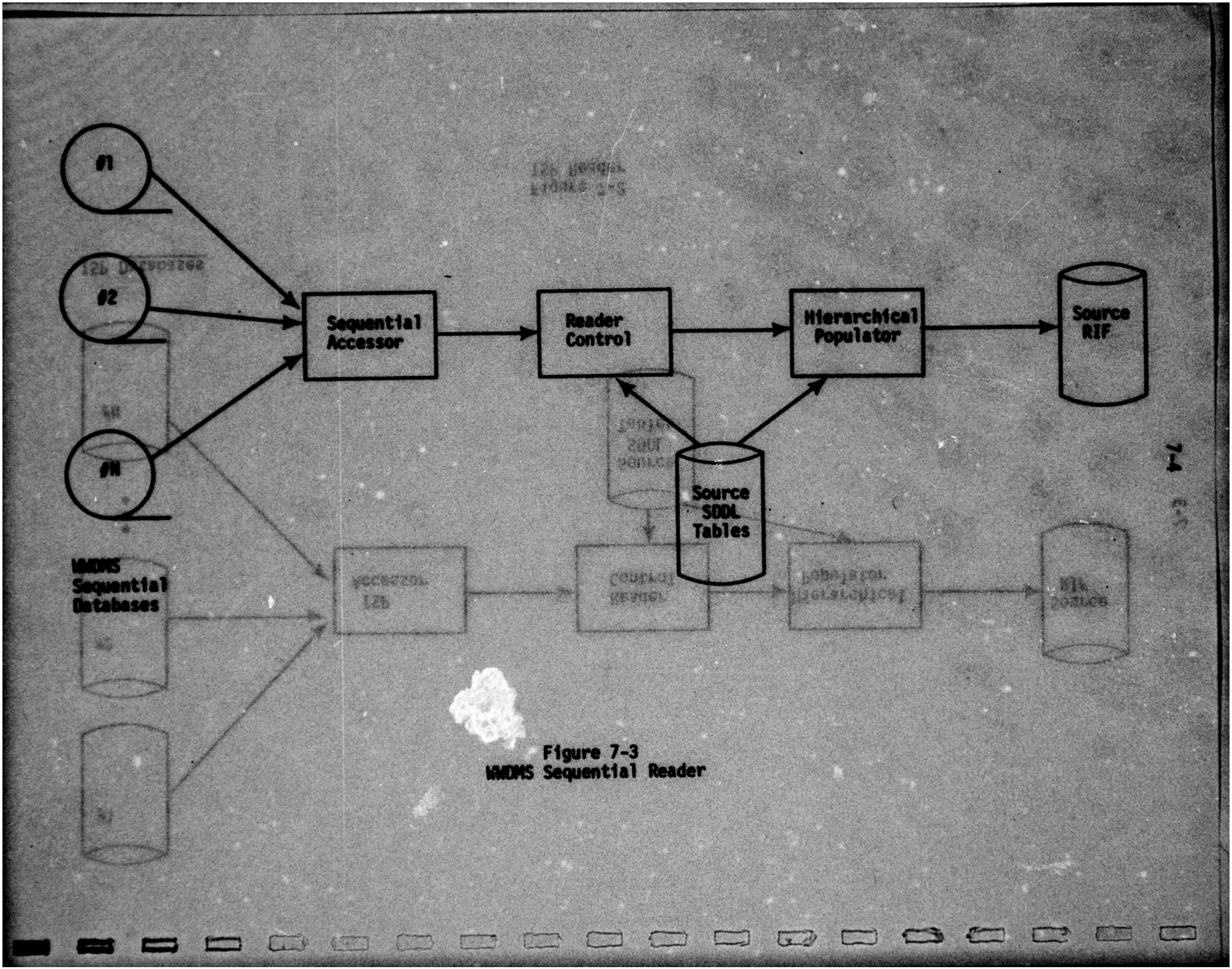


Figure 7-3
VMDMS Sequential Reader

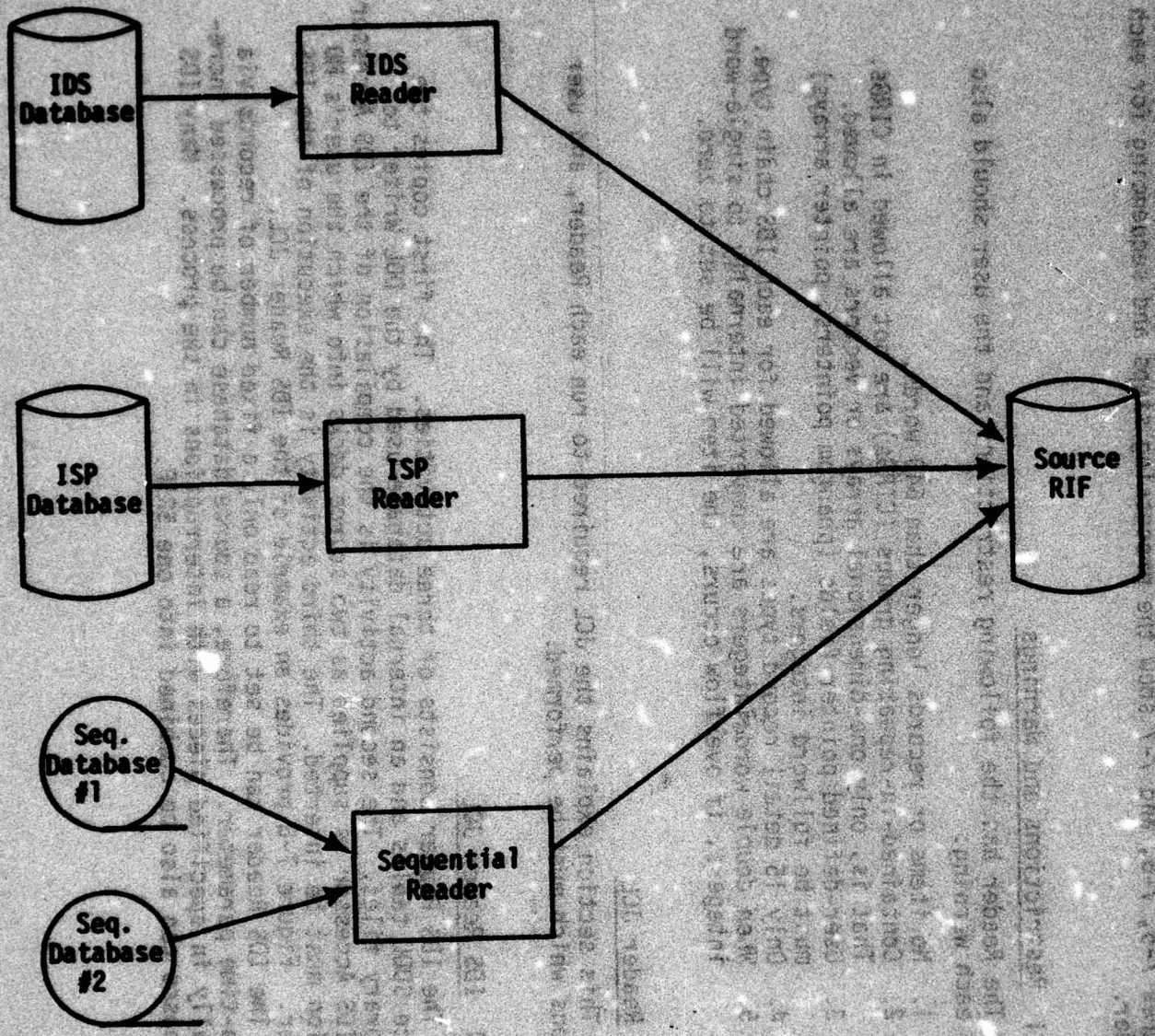


Figure 7-4
Reader Example

The second step is the major section of the Reader. Each record in the source database is read and moved to SRIF. Data conversion is performed if necessary. The Populator then links the record in the SRIF into all of its sets. This process continues until either the database has been entirely read or a user-specified number of records has been read. The last step, wrapup, involves dumping a sample of the SRIF, dumping the DRT (if IDS Reader) and linking match-key relations (if this is the last source database). Figures 7-5, 7-6, and 7-7 show the processing steps and sequencing for each Reader.

7.2.1 Restrictions and Warnings

The Reader has the following restrictions and the user should also note each warning:

1. No items or records longer than 640 words.
2. Contained-in-repeating groups (CIRGs) are not allowed in CIRGs. That is, only one-dimensional arrays or vectors are allowed.
3. User-defined pointer fields (phantom pointers, pointer arrays) must be fullword integers.
4. Only 15 detail record types are allowed for each IDS chain type.
5. When double word integers are converted internally to single-word integers, if overflow occurs, the item will be set to zero.

7.3 Reader JCL

This section explains the JCL required to run each Reader, and user actions which must be performed.

7.3.1 IDS Reader JCL

The IDS Reader consists of three activities. The first copies the source SDDL tables and an internal database used by the DDL Writer to temporary files. The second activity is the compilation of the IDS Accessor. The IDS Accessor is supplied as two source decks into which the user's MD section must be inserted. The third activity is the execution of the IDS Reader. Figure 7-8 provides an example of the IDS Reader JCL.

The IDS Reader can be set to read only a fixed number of records via a run-time parameter. Therefore, a source database can be processed incrementally in specified pieces with interruptions in the process. Many IDS databases can also be combined into one SRIF.

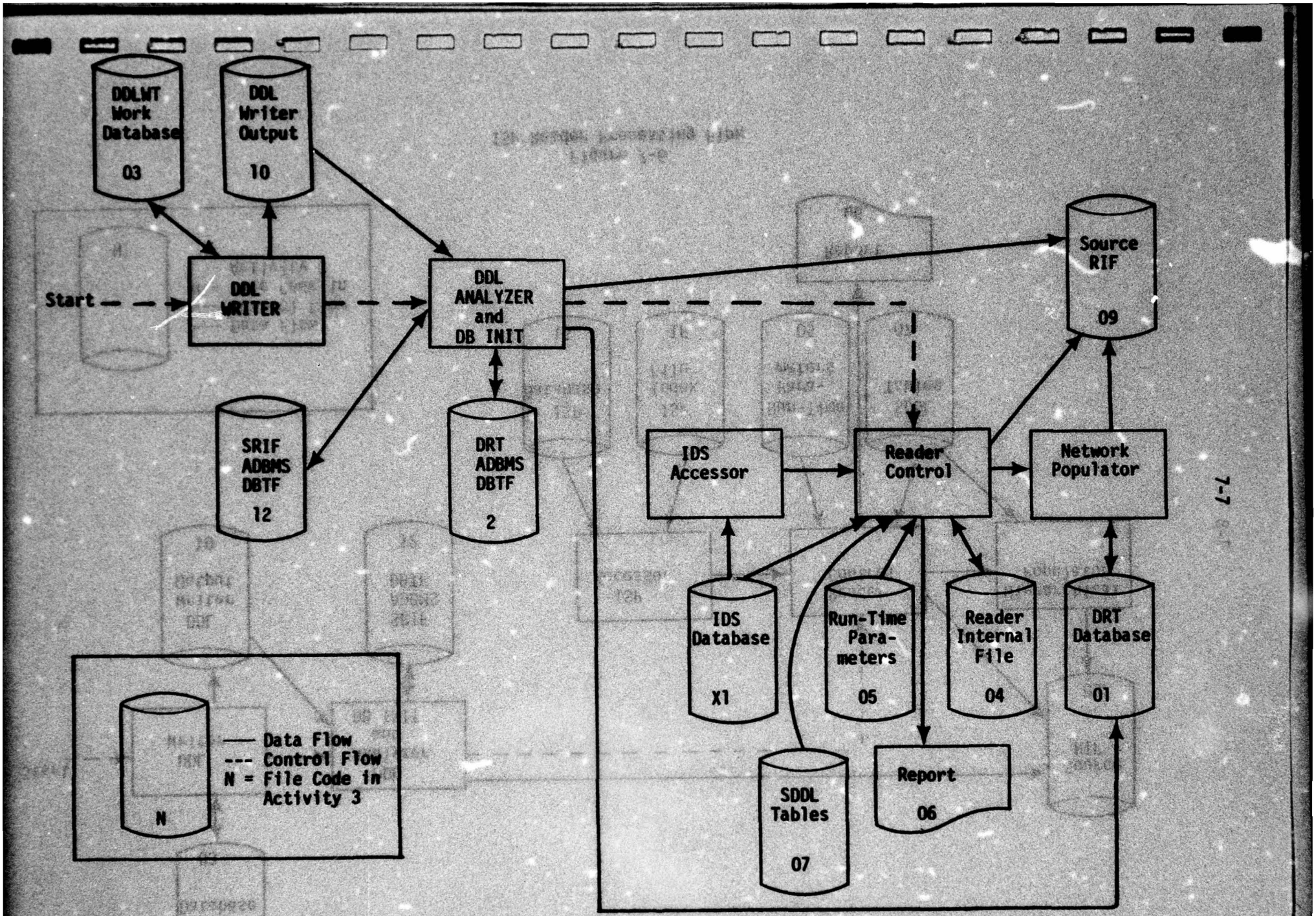


Figure 7-5
IDS Reader Processing Flow

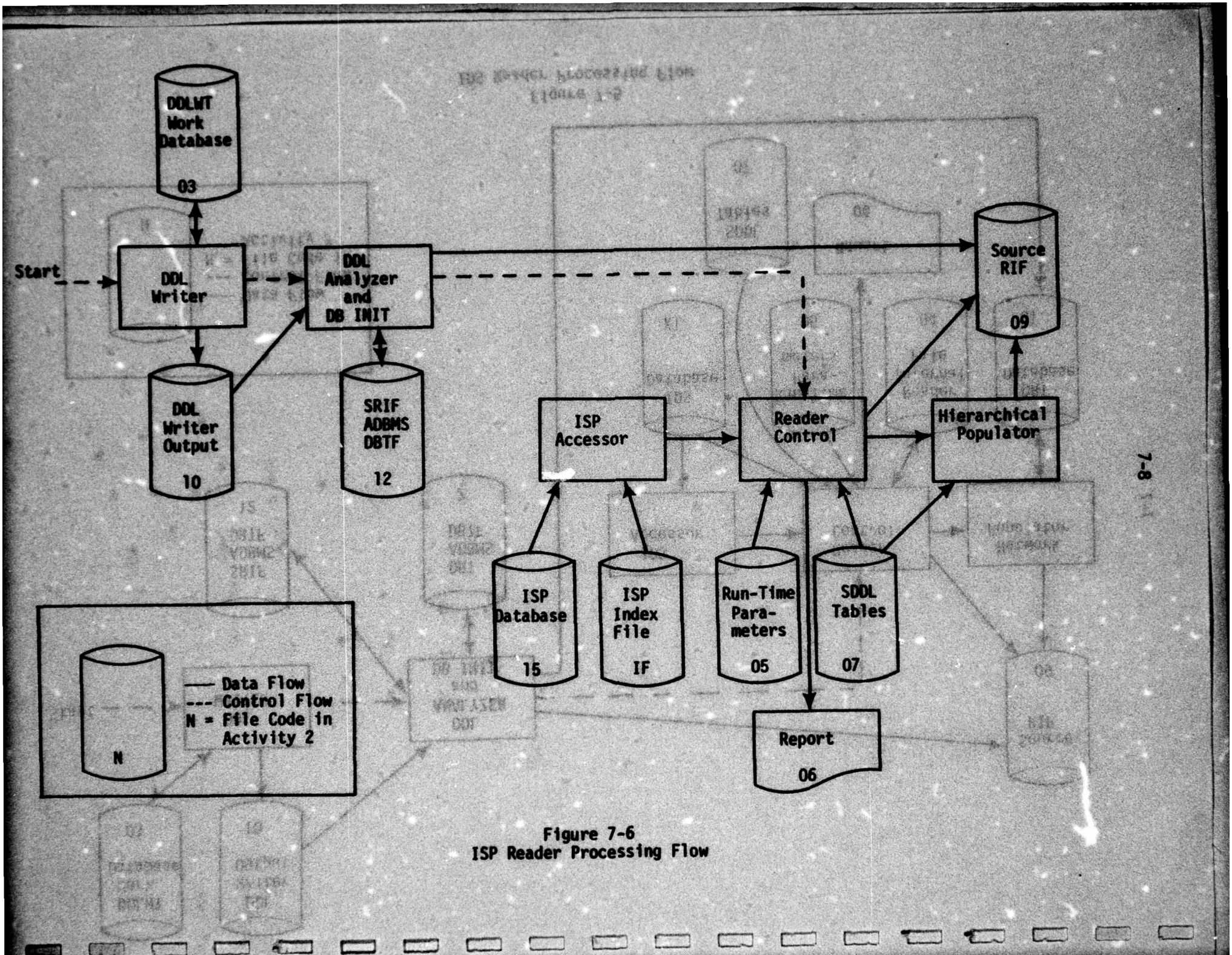


Figure 7-6
ISP Reader Processing Flow

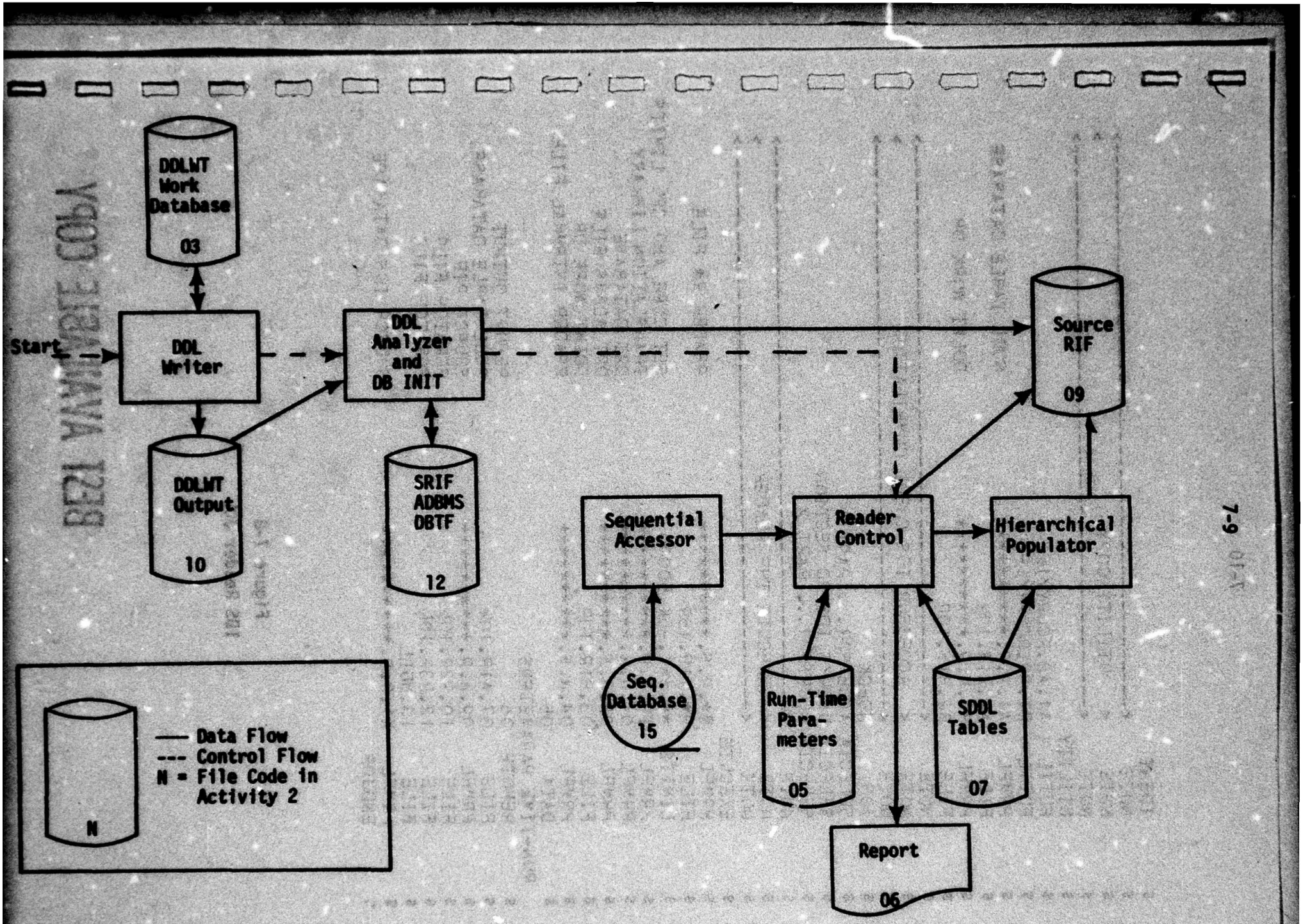


Figure 7-7
Sequential Reader Processing Flow

```
10      $       IDENT
20      $       NOTE    <------------------------------------------------>
30      $       NOTE    <   UTILITY COPY                                 >
40      $       NOTE    <------------------------------------------------>
50      $       UTILITY
60      $       FUTIL   A1.A2.RCOPY/1F/
70      $       FUTIL   B1.B2.RCOPY/1F/
80      $       PRMFL   A1.R.R.**********          SDDL TABLE DATABASE
90      $       FILE    A2.A1S.10R
100     $       PRMFL   B1.R.R.**********          DDLWT WORK DR
110     $       FILE    B2.B1S.10R
120     $       NOTE    <------------------------------------------------>
130     $       NOTE    <  ACCESSOR IDS-COBOL COMPILATION                >
140     $       NOTE    <------------------------------------------------>
150     $       IDS     NDECK
160     $       SELECTA ACCESSOR... PART 1
170     $       SELECTA USER'S IDS MD SECTION
180     $       SELECTA ACCESSOR... PART 2
190     $       NOTE    <------------------------------------------------>
200     $       NOTE    <  EXECUTE THE READER                            >
210     $       NOTE    <------------------------------------------------>
220     $       EXECUTE
230     $       PRMFL   R*.R.S.**********          READER R* FILE
        $       FILE    H*.Y1R.15R
        $       LIMITS  **,**K.-5K,50000          SET CORE AND CPU LIMITS
        $       PRMFL   TR.R.R.**********         TRANSLATION LIBRARY
        $       PRMFL   01.W.R.**********         DRT DATABASE
280     $       PRMFL   02.W.S.**********         DRT ADBMS FILE
290     $       FILE    03.B1R.10R                DDLWT WORK DR
300     $       PRMFL   04.W.S.**********         READER INTERNAL FILE
310     $       DATA    05
320         RUN-TIME PARAMETERS
330     $       REMOTE  06                        REPORT OUTPUT
340     $       FILE    07.A1R.10R                SDDL TABLE DATABASE
350     $       PRMFL   09.W.R.**********         SOURCE RIF
360     $       FILE    10.R2R.10L                SCRATCH FILE
370     $       FILE    12.R3R.10L                SCRATCH FILE
380     $       FILE    13.NULL
390     $       PRMFL   X1.R.R.**********         SOURCE IDS DATABASE
400     $       ENDJOB
```

Figure 7-8
IDS Reader JCL

BEST AVAILABLE COPY

7.3.2 ISP Reader JCL

```
10      $       IDENT
20      $       NOTE    <-------------------------------------------->
30      $       NOTE    <  UTILITY COPY                               >
40      $       NOTE    <-------------------------------------------->
50      $       UTILITY
60      $       FUTIL   A1.A2.RCOPY/1F/
70      $       FUTIL   B1.B2.RCOPY/1F/
80      $       PRMFL   A1.R.R.**********          SDDL TABLE DATABASE
90      $       FILE    A2.A1S.1OR
100     $       PRMFL   B1.R.R.**********          DDLWT WORK DB
110     $       FILE    B2.B1S.1OR
120     $       NOTE    <-------------------------------------------->
130     $       NOTE    <  EXECUTE THE READER                         >
140     $       NOTE    <-------------------------------------------->
150     $       EXECUTE
160     $       PRMFL   R*.R.S.**********          READER R* FILE
170     $       FILE    H*.Y1R.15R
180     $       LIMITS  **.**K.-5K.50000           SET CORE AND CPU LIMITS
190     $       PRMFL   TR.R.R.**********          TRANSLATION LIBRARY
200     $       FILE    03.B1R.1OR                 DDLWT WORK DB
210     $       DATA    05
220       RUN-TIME PARAMETERS
230     $       REMOTE  06                         REPORT OUTPUT
240     $       FILE    07.A1R.1OR                 SDDL TABLE DATABASE
250     $       PRMFL   09.W.R.**********          SOURCE RIF
260     $       FILE    10.R2R.10L                 SCRATCH FILE
270     $       FILE    12.R3R.10L                 SCRATCH FILE
280     $       FILE    13.NULL
290     $       PRMFL   15.R.R.**********          SOURCE ISP.DATA BASE
300     $       PRMFL   1F.R.R.**********          ISP INDEX FILE
310     $       DATA    DC                         ISP DATA CARDS
320     ISP     INDEX   FC=1F
330     ISP     DATA    FC=15
340     ISP     RECORD  **********                 SET RECORD PARAMETERS
350     $       ENDJOB
```

Figure 7-9

ISP Reader JCL

The ISP Reader consists of two activities. The first activity copies the SDDL tables and an internal database used by the DDL Writer to temporary files. The second activity is the execution of the ISP Reader. Figure 7-9 indicates the JCL required to run the ISP Reader.

BEST AVAILABLE COPY

7.3.3 WNDMS Sequential Reader JCL

```
10      $       IDENT
20      $       NOTE    <------------------------------------------------>
30      $       NOTE    <   UTILITY COPY                                 >
40      $       NOTE    <------------------------------------------------>
50      $       UTILITY
60      $       FUTIL   A1,A2,RCOPY/1F/
70      $       FUTIL   B1,B2,RCOPY/1F/
80      $       PRMFL   A1,R,R,**********          SDDL TABLE DATABASE
90      $       FILE    A2,A1S,10R
100     $       PRMFL   B1,R,R,**********          DDLWT WORK DB
110     $       FILE    B2,B1S,10R
120     $       NOTE    <------------------------------------------------>
130     $       NOTE    <  EXECUTE THE READER                            >
140     $       NOTE    <------------------------------------------------>
150     $       EXECUTE
160     $       PRMFL   R*,R,S,**********          READER R* FILE
170     $       FILE    H*,Y1R,15R
180     $       LIMITS  **,**K,-5K,50000           SET CORE AND CPU LIMITS
190     $       PRMFL   TR,R,R,**********          TRANSLATION LIBRARY
200     $       FILE    03,B1R,10R                 DDLWT WORK DB
210     $       DATA    05
220         RUN-TIME PARAMETERS
230     $       REMOTE  06                         REPORT OUTPUT
240     $       FILE    07,A1R,10R                 SDDL TABLE DATABASE
250     $       PRMFL   09,W,R,**********          SOURCE RIF
260     $       FILE    10,R2R,10L                 SCRATCH FILE
270     $       FILE    12,R3R,10L                 SCRATCH FILE
280     $       FILE    13,NULL
290     $       PRMFL   15,R,S,**********          SOURCE SEQ. DATABASE
300     $       ENDJOB
```

Figure 7-10

Sequential Reader JCL

The Sequential Reader consists of two activities. The first activity copies the SDDL tables and an internal database used by the DDL Writer to temporary files. The second activity executes the Sequential Reader. Figure 7-10 indicates the JCL for the Sequential Reader.

7.4 Files and Reports

An explanation of the files and output for the IDS Reader is given in Section 7.5. Sections 7.6 and 7.7 explain the files and output for the ISP and sequential Readers.

BEST AVAILABLE COPY

BEST AVAILABLE COPY

7.5 IDS Reader Files and Reports

```
10      $       IDENT
20      $       NOTE    <-------------------------------------------------->
30      $       NOTE    <   UTILITY COPY                                    >
40      $       NOTE    <-------------------------------------------------->
50      $       UTILITY
60      $       FUTIL   A1.A2.RCOPY/1F/
70      $       FUTIL   B1.B2.RCOPY/1F/
80      $       PRMFL   A1.R.R.**********          SDDL TABLE DATABASE
90      $       FILE    A2.A1S.IOR
100     $       PRMFL   B1.R.R.**********          DDLWT WORK DB
110     $       FILE    B2.B1S.IOR
```

Figure 7-11

Activity 1

7.5.1 Activity 1

The first activity of the IDS Reader copies the source SDDL tables and an internal database used by the DDL Writer to temporary files. Line 80 (in Figure 7-11) should refer to the file containing the source SDDL tables. Line 100 should refer to the DDL writer's work database that was brought off the system generation tape (see Appendix G). The user should check report code 53 to be sure the first activity terminated normally.

7.5.2 Activity 2

```
120     $       NOTE    <-------------------------------------------------->
130     $       NOTE    <  ACCESSOR IDS-COBOL COMPILATION                   >
140     $       NOTE    <-------------------------------------------------->
150     $       IDS     NDECK
160     $       SELECTA ACCESSOR... PART 1
170     $       SELECTA USER'S IDS MD SECTION
180     $       SELECTA ACCESSOR... PART 2
```

Figure 7-12

Activity 2

The second activity compiles the IDS Accessor. The two sections of Accessor source code should be brought off the system generation tape (see Appendix G) and saved in permanent files. Lines 160 and 180 (in Figure 7-12) should refer to those files. The MD section for the database to be read should be inserted in line 170. The MD section should begin with MD FILE IS...not IDS-SECTION. The following words should not be included in the user's MD section:

ACC-C-ADDR	ACC-ERROR-VALUE
ACC-CHECK-FOR-PAGE-HEADER	ACC-FIRST-TIME
ACC-DEFINE-SYMBOLS	ACC-IERROR
ACC-DELETE-SWITCH	ACC-LEN
ACC-END-DB	ACC-OPEN-DB
ACC-EOF	ACC-REC-TYP

BEST AVAILABLE COPY

BEST AVAILABLE COPY

ACC-REF-COD	ACC-TOTAL-REC
ACC-RETRIEVE-RECORD	ACC-TOTAL-RECORDS
ACC-RETURN-PARA	ACC-TOTAL-STRING
ACC-SET-UP-REF-CODES	ACC-TR
	ACC-W-ADDR

The MD section, not the Accessor code, should be changed if any of the above words appear in the user's MD section. Report code 74 should be checked for IDS or COBOL compilation errors. Errors due to name conflict should be corrected and the Reader can be run.

7.5.3 Activity 3

```
190     $       NOTE    <-------------------------------------------------->
200     $       NOTE    <  EXECUTE THE READER                              >
210     $       NOTE    <-------------------------------------------------->
220     $       EXECUTE
230     $       PRMFL   R*.R.S.**********          READER R* FILE
240     $       FILE    H*.Y1R.15P
250     $       LIMITS  **.**K.-5K.50000           SET CORE AND CPU LIMITS
260     $       PRMFL   TR.R.R.**********          TRANSLATION LIBRARY
270     $       PRMFL   01.R.R.**********          DRT DATABASE
280     $       PRMFL   02.R.S.**********          DRT ADMS FILE
290     $       FILE    03.R1R.1OR                 DDLWT WORK DB
300     $       PRMFL   04.R.S.**********          READER INTERNAL FILE
310     $       DATA    05
320         RUN-TIME PARAMETERS
330     $       REMOTE  06                         REPORT OUTPUT
340     $       FILE    07.A1R.1OR                 SDDL TABLE DATABASE
350     $       PRMFL   09.R.R.**********          SOURCE DIF
360     $       FILE    10.R2R.1OL                 SCRATCH FILE
370     $       FILE    12.R3R.1OL                 SCRATCH FILE
380     $       FILE    13.NULL
390     $       PRMFL   X1.P.R.**********          SOURCE IDS DATABASE
400     $       ENDJOB
```

Figure 7-13
Activity 3

The following lines should be set in Figure 7-13 as follows:

Line	Contents
230	This should be the IDS Reader R* file to be used. It should be saved in a permanent file created from the system generation tape (Appendix G).
250	The core limit should be set to the size given for the IDS Reader in Appendix G. The CPU limit should be set so that the Reader can finish in the allotted time. Assume that the Reader can process approximately 6000 IDS records per CPU hour.

BEST_AVAILABLE COPY

260 The Translation Library is a library of low-level routines essential to the Reader. It can be found on the system generation tape (Appendix G).

270 The user should create a random file which is about twenty-five percent the size of the source IDS database to be used as the DRT database. That file should be referenced here.

280 The DRT ADBMS DBTF should be copied off the system generation tape to a permanent file (1 llink, sequential).

300 The IDS Reader's Internal file should be copied off the system generation tape to a permanent file (1 llink, sequential).

320 The run-time parameters should be set here. There are five run-time parameters:

1. The word "FIRST" should be included on this line if and only if this is the first time the Reader has accessed the source database.
2. The word "LAST" should be included on this line if and only if this is the last run on the source database. Note that if there is only one run of the Reader "FIRST" and "LAST" both should be used.
3. The word "DEBUG" should be included on this line if and only if a list of IDS reference codes and corresponding ADBMS keys is desired. For each set instance in each record instance, the IDS reference codes and set type will be printed. Note that this could potentially produce an extremely large amount of output.
4. The name of the database must be included on this line in the form:

   NAME=database-name

   where database-name is the name of the database and its first character immediately follows the equals sign.
5. The number of records to be retrieved from the IDS database should be included here in the form:

   RECORDS=n or
   RECORDS=ALL

   where the first n records will be retrieved or ALL records are to be retrieved. Note that the IDS Reader can read a database of 100,000 records by running the Reader ten times with RECORDS =10000.

350 The source RIF should be a random, permanent file. It should be about twice the size of all source databases.

390 The source IDS database should have filecode X1. Subfiles should have filecodes X2, X3, etc.

7.5.4 IDS Reader Output and DRT Dump

All Reader output which is of any value to the user can be found in report code 06. An example of the first page of output is shown in Figure 7-14. After the page header, the run-time parameters are printed as they appeared in the run-time parameter file. The next four lines are initialization messages from the ADBMS system. The first initialization message corresponds to the SRIF and the second line corresponds to the DRT database. (Note that a page is 1024 words). If there were no errors during the Reader execution, the number of IDS records retrieved and the number of logically deleted IDS records which were physically deleted in the SRIF will be printed. If there were errors in the Reader execution, error messages will be written after the database initialization message.

Beginning on the second page of output will be the final dump of the DRT database. The DRT dump contains information concerning every chain-type relation in the source IDS database(s). For each set in the SRIF there will be a line of output with the name of the set (RELATION), the set type (TYPE), the master and detail records (PARENT , DEPENDENT), the sibling set type (SIBLING), and the displacement and length of the pointer fields in the master and detail (OWNDIS, OWNLEN, MEMDIS, MEMLEN). The set type (TYPE) can be either an IDS chain (IDSCHN) or a user's phantom pointer (PHNCHN). The sibling set type (SIBLING) can be disregarded by the user. The length and displacement of the pointer fields should be checked for correctness.

Since the DRT dump can take on many different forms the following general tips should help the user determine whether the Reader ran correctly. These tips apply only when the Reader is run for the LAST time. For all other runs, the DRT dump can be ignored.

1. For IDSCHN-type relations, only the line of output with the relation name, etc., should appear. (See Figure 7-15). If the relation header line is followed by a line with FIRST, NEXT, PARENT, etc., some sort of error has occurred. The problem could be in the source database(s) or the Reader. Data Translation personnel should be contacted to pinpoint the error.
2. For PHNCHN-type relations, the line of output with the relation name, etc., should be followed by FIRST, NEXT, PARENT, etc., if there were any instances of that relation in the source database. Figure 7-16 shows an example of this type of output. Under the word PARTYP, REAL should be the only word which appears. If the word SURG is present, an error has occurred and Data Translation personnel should be contacted. Under IDSKEY is the IDS reference code of the owner of the set instance and under NUMMEM is the number of members of the set instance. A few of the set instances should be checked in the source database to be sure they are correct.
3. If any IDS reference code field (FIRST, NEXT, IDSKEY) is not a valid IDS reference code (maybe it is blanks or other characters), the user should check the OWNDIS, OWNLEN, MEMDIS and MEMLEN for the correct values. If an incorrect value appears, the IDS MD section should be corrected, the SDDL tables should be recreated and the Reader should be rerun.

```
   THE UNIVERSITY OF MICHIGAN IDS READER(VERSION IIA, RELEASE 2) 12/14/76      21.579

RUN-TIME PARAMETERS--->  NAME=OLD-PRESIDENTIAL  FIRST LAST RECORDS=ALL                <---

        BLOCK DATA FILE WRITTEN

        DATA BASE TABLE FILE WRITTEN

 DATA BASE INITIALIZED WITH    25 PAGES.

 DATA BASE INITIALIZED WITH    18 PAGES.

THE TOTAL NUMBER OF RECORDS RETRIEVED IS--->        371
THE NUMBER OF LOGICALLY DELETED RECORDS WHICH WERE PHYSICALLY DELETED IN THE SRIF IS--->          0
```

Figure 7-14

```
RELATION-->C-PCL  TYPE-->IDSCHN PARENT-->CONGRE SIBLING-->        DEPENDENT-->PRES-C
RELATION-->P-A    TYPE-->IDSCHN PARENT-->PRESID SIBLING-->        DEPENDENT-->ADMINI
RELATION-->P-E    TYPE-->IDSCHN PARENT-->PRESID SIBLING-->        DEPENDENT-->ELECTI
RELATION-->P-PCL  TYPE-->IDSCHN PARENT-->PRESID SIBLING-->        DEPENDENT-->PRES-C
RELATION-->S-P    TYPE-->IDSCHN PARENT-->STATES SIBLING-->        DEPENDENT-->PRESID
```

Figure 7-15

THE UNIVERSITY OF MICHIGAN IDS READER(VERSION IIA, RELEASE 2) 12/14/76 22.00

DEFERRED REFERENCE TABLE (DRT) DUMP

CLOSE OF DRT

RELATION-->ADMITT FIRST	TYPE-->PHNCHN NEXT	PARENT-->STATES PARENT	PARTYP	SIBLING--> IDSKEY	NUMMEM	DEPENDENT-->STAT
000000000000	000000000000	000010001246	REAL	000000001601	1	
000000000000	000000000000	000010001323	REAL	000000001602	1	
000000000000	000000000000	000004000437	REAL	000000001603	1	
000000000000	000000000000	000010001635	REAL	000000001604	1	
000000000000	000000000000	000002001361	REAL	000000001701	0	
000000000000	000000000000	000011000164	REAL	000000001702	1	
000000000000	000000000000	000011000241	REAL	000000001703	1	
000000000000	000000000000	000002000633	REAL	000000001704	1	
000000000000	000000000000	000011000410	REAL	000000002001	1	
000000000000	000000000000	000011000465	REAL	000000002002	1	
000000000000	000000000000	000011000631	REAL	000000002003	1	
000000000000	000000000000	000011000673	REAL	000000002101	0	
000000000000	000000000000	000003001532	REAL	000000002301	0	
000000000000	000000000000	000011001144	REAL	000000002302	1	
000000000000	000000000000	000011001247	REAL	000000002303	1	
000000000000	000000000000	000003000664	REAL	000000002601	1	
000000000000	000000000000	000011001416	REAL	000000002602	1	
000000000000	000000000000	000011001534	REAL	000000002603	1	
000000000000	000000000000	000002001015	REAL	000000002604	1	
000000000000	000000000000	000011001731	REAL	000000002701	1	
000000000000	000000000000	000012000016	REAL	000000002702	1	
000000000000	000000000000	000012000060	REAL	000000002703	1	
000000000000	000000000000	000012000122	REAL	000000002704	1	
000000000000	000000000000	000012000177	REAL	000000003001	1	
000000000000	000000000000	000012000267	REAL	000000003002	1	
000000000000	000000000000	000012000344	REAL	000000003003	1	
000000000000	000000000000	000012000434	REAL	000000003101	0	
000000000000	000000000000	000012000476	REAL	000000003102	0	
000000000000	000000000000	000012000601	REAL	000000003103	0	
000000000000	000000000000	000012001037	REAL	000000003104	1	
000000000000	000000000000	000012001127	REAL	000000003201	1	
000000000000	000000000000	000012001423	REAL	000000003202	1	
000000000000	000000000000	000012001465	REAL	000000003203	0	
000000000000	000000000000	000004000640	REAL	000000003204	1	
000000000000	000000000000	000012001647	REAL	000000003301	1	
000000000000	000000000000	000002001177	REAL	000000003302	1	
000000000000	000000000000	000013000044	REAL	000000003303	0	
000000000000	000000000000	000003001350	REAL	000000003304	1	
000000000000	000000000000	000013000152	REAL	000000003401	1	
000000000000	000000000000	000013000214	REAL	000000003402	1	
000000000000	000000000000	000013000256	REAL	000000003403	1	
000000000000	000000000000	000013000320	REAL	000000003404	0	
000000000000	000000000000	000005000654	REAL	000000003501	1	
000000000000	000000000000	000013000604	REAL	000000003502	0	
000000000000	000000000000	000013000661	REAL	000000003503	1	
000000000000	000000000000	000013000736	REAL	000000003504	0	
000000000000	000000000000	000013001044	REAL	000000003601	1	
000000000000	000000000000	000013001134	REAL	000000003602	1	
000000000000	000000000000	000013001237	REAL	000000003603	0	
000000000000	000000000000	000004000123	REAL	000000003604	0	

Figure 7-16
DRT Dump

```
THE UNIVERSITY OF MICHIGAN DBMS-11 ADMINISTRATOR VERSION I RELEASE 12.1/7 3.9

SYSTEM OWNED SET SYS115 ********************
THE NUMBER OF MEMBERS IS--->        50

  SOURCE RIF RECORD TYPE IS ----> STATES    ADBMS DATABASE KEY FOR THIS RECORD IS ----> 000010013246

  SET POINTERS FOR THIS RECORD ARE...
MEMBER OF SYS115      PRIOR-->000000000000 OWNER-->000002000003 NEXT-->000010001367
OWNER  OF ADMITT      FIRST-->000000000000 LAST -->000000000000 # OF MEMBERS-->          0
MEMBER OF CALC-S      PRIOR-->000000000000 OWNER-->000002000003 NEXT-->000010001367
OWNER  OF S-P         FIRST-->000000000000 LAST -->000000000000 # OF MEMBERS-->          0
MEMBER OF STATED      PRIOR-->000000000000 OWNER-->000010001323 NEXT-->000004000123
OWNER  OF STATEA      FIRST-->000010001310 LAST -->000010001310 # OF MEMBERS-->          1

  DATA ITEMS FOR THIS RECORD ARE...
YEAR-A *1876*
CAPITA *DENVER    *
AREA-S *    104247*
AREA-R * 8*
POPULA *   2048000*
POP-RA *30*
ELECTO * 6*
ADMIN- 7452565520
ADMINR *   *
STATE- *COLORADO  *

  SOURCE RIF RECORD TYPE IS ----> STATES    ADBMS DATABASE KEY FOR THIS RECORD IS ----> 000004020123

  SET POINTERS FOR THIS RECORD ARE...
MEMBER OF SYS115      PRIOR-->000013000371 OWNER-->000002000003 NEXT-->000000000000
OWNER  OF ADMITT      FIRST-->000000000000 LAST -->000000000000 # OF MEMBERS-->          0
MEMBER OF CALC-S      PRIOR-->000013000371 OWNER-->000002000003 NEXT-->000000000000
OWNER  OF S-P         FIRST-->000002000305 LAST -->000004001034 # OF MEMBERS-->          2
MEMBER OF STATED      PRIOR-->000010001246 OWNER-->000010001323 NEXT-->000013000371
OWNER  OF STATEA      FIRST-->000013000522 LAST -->000013000563 # OF MEMBERS-->          4

  DATA ITEMS FOR THIS RECORD ARE...
YEAR-A *1789*
CAPITA *RALEIGH   *
AREA-S *     57712*
AREA-R *24*
POPULA *   5135000*
POP-RA *11*
ELECTO *13*
ADMIN-       2057
ADMINR *A23*
```

Figure 7-17
SRIF Dump

```
KEY=000002000077  IDS REF=000000000101 PRESID
  PRESID  P-A     000000000101 000000003701   1   0
  PRESID  P-E     000000000101 000000001301   1   0
  PRESID  P-PCL   000000000101 000000005312   1   0
  PRESID  S-P     000000000101 000000003203   2   0
K[illegible]002000305  IDS REF=000000000102 PRESID
          [illegible]  000000000102 000000003714   1   0
          [illegible]  000000000102 000000001003   1   0
          -PCL  000000000102 000000005345   1   0
          -P    000000000102 000000003604   2   0
KEY=000002000501  IDS REF=000000000103 PRESID
  PRESID  P-A     000000000103 000000004102   1   0
  PRESID  P-E     000000000103 000000000103   1   0
  PRESID  P-PCL   000000000103 000000005372   1   0
  PRESID  S-P     000000000103 000000001703   2   0
KEY=000002000675  IDS REF=000000000104 PRESID
  PRESID  P-A     000000000104 000000004204   1   0
  PRESID  P-E     000000000104 000000001010   1   0
  PRESID  P-PCL   000000000104 000000005426   1   0
  PRESID  S-P     000000000104 000000002603   2   0
KEY=000002001057  IDS REF=000000000105 PRESID
  PRESID  P-A     000000000105 000000004205   1   0
  PRESID  P-E     000000000105 000000001207   1   0
  PRESID  P-PCL   000000000105 000000005435   1   0
  PRESID  S-P     000000000105 000000000205   2   0
KEY=000002001241  IDS REF=000000000106 PRESID
  PRESID  P-A     000000000106 000000004213   1   0
  PRESID  P-E     000000000106 000000001013   1   0
  PRESID  P-PCL   000000000106 000000005502   1   0
  PRESID  S-P     000000000106 000000001604   2   0
KEY=000002001423  IDS REF=000000000201 PRESID
  PRESID  P-A     000000000201 000000003707   1   0
  PRESID  P-E     000000000201 000000001001   1   0
  PRESID  P-PCL   000000000201 000000005336   1   0
  PRESID  S-P     000000000201 000000003504   2   0
KEY=000002001631  IDS REF=000000000202 PRESID
  PRESID  P-A     000000000202 000000003711   1   0
  PRESID  P-E     000000000202 000000001002   1   0
  PRESID  P-PCL   000000000202 000000005340   1   0
  PRESID  S-P     000000000202 000000003103   2   0
KEY=000003000045  IDS REF=000000000203 PRESID
  PRESID  P-A     000000000203 000000004014   1   0
  PRESID  P-E     000000000203 000000001205   1   0
  PRESID  P-PCL   000000000203 000000005367   1   0
  PRESID  S-P     000000000203 000000000603   2   0
```

Figure 7-18

DEBUG Output

7.5.5 Source RIF Dump

After the DRT dump, a small portion of the SRIF is dumped. For each SYSTEM-owned set in the SRIF, the first and last member of the set is dumped. An example of this output is shown in Figure 7-17. After the page header, the name of the set and the number of members are printed. The name of the member record and the ADBMS key for that record instance are then printed. The user can generally ignore the list of set pointers for the record. The list of data items should be closely checked for errors such as missing data and data with unreasonable values. For each data item, the name of the field is written and then its value is printed. Character data items are delimited by asterisks whereas integer and floating point data are not delimited. If an error is found in any item, the user should determine whether the problem is in the IDS database or the Reader, and then correct it. Note that some records may be dumped twice if they are members of CALC sets.

7.5.6 Debug Output

When the user specifies DEBUG as a run-time parameter, additional output will be written on report code 06. The DRT will be dumped after it has been initialized, and for each record instance in the source database(s), the following output will be printed:

1. The IDS reference code of the record (IDS REF=).
2. The ADBMS key of the logically equivalent record in the SRIF (KEY=).
3. For each set instance in the record, a line will be printed with the name of the record, the name of the set, the IDS reference code of the current record, the IDS reference code of the NEXT record in this set instance, a flag (=2 if member in the set, =1 if owner of the set) and the deletion flag (=1 if logically deleted).

An example of the debug output is shown in Figure 7-18. The user should note that the DEBUG feature can produce a considerable amount of output for a large database.

7.6 ISP Reader Files and Reports

```
10      $       IDENT
20      $       NOTE    <-----------------------
```

7.6.1 Activity 1

The first activity of the ISP Reader copies the source SDDL tables and an internal database used by the DDL Writer to temporary files. Line 80 (in Figure 7-19) should refer to the file containing the source SDDL tables. Line 100 should refer to the DDL Writer's work database that was brought off the system generation tape (see Appendix G). The user should check report code 53 to be sure the first activity terminated normally.

7.6.2 Activity 2

```
120     $       NOTE    <-------------------------------------------------->
130     $       NOTE    <  EXECUTE THE READER                              >
140     $       NOTE    <-------------------------------------------------->
150     $       EXECUTE
160     $       PRMFL   R*.R.S.**********          READER R* FILE
170     $       FILE    H*.YIR.15R
180     $       LIMITS  **.**K.-5K.50000           SET CORE AND CPU LIMITS
190     $       PRMFL   IR.R.R.**********          TRANSLATION LIBRARY
200     $       FILE    03.RIR.10R                 DDLWT WORK DB
210     $       DATA    05
220         RUN-TIME PARAMETERS
230     $       REMOTE  06                         REPORT OUTPUT
240     $       FILE    07.AIR.10R                 SDDL TABLE DATABASE
250     $       PRMFL   09.W.R.**********          SOURCE RIF
260     $       FILE    10.R2R.10L                 SCRATCH FILE
270     $       FILE    12.R3R.10L                 SCRATCH FILE
280     $       FILE    13.NULL
290     $       PRMFL   15.P.R.**********          SOURCE ISP DATA BASE
300     $       PRMFL   IF.R.R.**********          ISP INDEX FILE
310     $       DATA    DC                         ISP DATA CARDS
320     ISP     INDEX   FC=IF
330     ISP     DATA    FC=15
340     ISP     RECORD  **********                 SET RECORD PARAMETERS
350     $       ENDJOB
```

Figure 7-20

Activity 2

To run the ISP Reader the following lines should be set in Figure 7-20:

Line	Contents
160	This should be the ISP Reader R* file. It should be brought off the system generation tape (see Appendix G).
180	The core limit should be set to the size given for the ISP R* file in Appendix G. The CPU limit should be set so that the Reader can finish in the allotted time. Assume the Reader can process approximately 6000 records per hour.

BEST AVAILABLE COPY

190	The Translation Library is a library of low-level routines essential to the Reader. It can be found on the system generation tape (Appendix G).
220	The run-time parameters should be set here. There are three run-time parameters. 1. The word "FIRST" should be included on this line if and only if this is the first time the Reader has accessed the source database(s). 2. The word "LAST" should be included in this line if and only if this is the last run on the source database. Note that if there is only one run of the Reader, both "FIRST" and "LAST" should be used. 3. The name of database must be included on this line in the form: NAME=database-name where database-name is the name of the database and its first character immediately follows the equal sign.
250	The source RIF should be a random permanent file. It should be created about twice the size of all source databases.
290	The source ISP database should have filecode 15.
300	The ISP index file should have filecode IF.
320,330	These ISP data cards should be changed only if a non-standard (320 words) page size or commercial collating sequence is used in the ISP database.
340	The ISP parameters specifying record length, key size, and key offset should be set here.

7.6.3 ISP Reader Output

The output from the ISP Reader is on report code 06. The statistical report from the ISP system is on report code 73 but the user can ignore that output. An example of the first page of output is shown in Figure 7-21. After the page header, there is a line with the run-time parameters as they were listed in the run-time parameter file. The next three lines of output are initialization messages from the ADBMS system. The database which is initialized is the source RIF. If there were no errors during the execution of the Reader the next line should be the number of records retrieved from the ISP database. If there were errors during the run, the error messages will be printed after the initialization message.

The other output from the ISP Reader is a dump of a few records in the source RIF. Since this output is the same in the IDS Reader, the user should refer to Section 7.5.5 and Figure 7-17 for an explanation and sample of that output.

```
THE UNIVERSITY OF MICHIGAN ISP READER(VERSION IIA. RELEASE 2)  12/22/76     14.538
RUN-TIME PARAMETERS--->  NAME=PRES-IS  FIRST LAST                          <---
          BLOCK DATA FILE WRITTEN
          DATA BASE TABLE FILE WRITTEN
 DATA BASE INITIALIZED WITH    50 PAGES.
THE TOTAL NUMBER OF RECORDS RETRIEVED IS--->     3401
```

Figure 7-21

7.7 Sequential Reader Files and Reports

```
10      $       IDENT
20      $       NOTE    <-------------------------------------------------->
30      $       NOTE    <  UTILITY COPY                                     >
40      $       NOTE    <-------------------------------------------------->
50      $       UTILITY
60      $       FUTIL   A1.A2.RCOPY/1F/
70      $       FUTIL   B1.B2.RCOPY/1F/
80      $       PRMFL   A1.R.R.**********          SDDL TABLE DATABASE
90      $       FILE    A2.A1S.10R
100     $       PRMFL   B1.R.R.**********          DDLWT WORK DB
110     $       FILE    B2.B1S.10R
```

Figure 7-22
Activity 1

7.7.1 Activity 1

The first activity of the sequential Reader copies the source SDDL tables and an internal database used by the DDL Writer to temporary files. Line 80 (in Figure 7-22) should refer to the file containing the source SDDL tables. Line 100 should refer to the DDL Writer's work database that was brought off the system generation tape (see Appendix G). The user should check report code 53 to be sure the first activity terminated normally.

7.7.2 Activity 2

```
120     $       NOTE    <-------------------------------------------------->
130     $       NOTE    <  EXECUTE THE READER                               >
140     $       NOTE    <-------------------------------------------------->
150     $       EXECUTE
160     $       PRMFL   R*.R.S.**********          READER R* FILE
170     $       FILE    H*.Y1R.15R
180     $       LIMITS  **.**K.-5K.50000          SET CORE AND CPU LIMITS
190     $       PRMFL   TR.R.R.**********         TRANSLATION LIBRARY
200     $       FILE    03.B1R.10R                DDLWT WORK DB
210     $       DATA    05
220         RUN-TIME PARAMETERS
230     $       REMOTE  06                        REPORT OUTPUT
240     $       FILE    07.A1R.10R                SDDL TABLE DATABASE
250     $       PRMFL   09.W.R.**********         SOURCE RIF
260     $       FILE    10.R2R.10L                SCRATCH FILE
270     $       FILE    12.R3R.10L                SCRATCH FILE
280     $       FILE    13.NULL
290     $       PRMFL   15.R.S.**********         SOURCE SEQ. DATABASE
300     $       ENDJOB
```

Figure 7-23
Activity 2

To run the sequential Reader the following lines should be changed in Figure 7-23:

Line	Contents
160	This should be the sequential Reader R* file. It should be brought off the system generation tape (see Appendix G).
180	The core limit should be set to the size given for the sequential Reader in Appendix G. The CPU limit should be set so that the Reader can finish in the allotted time. Assume the Reader can process approximately 6000 records per hour.
190	The Translation Library is a library of low-level routines essential to the Reader. It should be brought off the system generation tape (see Appendix G).
220	The run-time parameters should be set here. There are three run-time parameters: 1. The word "FIRST" should be included on this line if and only if this is the first time the Reader has accessed the source database(s). 2. The word "LAST" should be included in this line if and only if this is the last run on the source database. Note that if there is only one run of the Reader, both "FIRST" and "LAST" should be used. 3. The name of the database must be included on this line in the form: NAME=database-name where database-name is the name of the database and its first character immediately follows the equal sign.
250	The source RIF should be a random permanent file. It should be created about twice the size of all source databases.
290	The source sequential database should have file code 15. If the database is on a tape, the LUD used for the tape drive should not conflict with any other LUDs used by the Reader.

7.7.3 Sequential Reader Output

The output from the sequential Reader is on report code 06. An example of the first page of output is shown in Figure 7-24. After the page header, there is a line with the run-time parameters as they appeared in the run-time parameter file. The next three lines of output are from the ADBMS system during the SRIF initialization process. If there were no errors during the execution of the Reader, the next line should be the number of records retrieved

```
    THE UNIVERSITY OF MICHIGAN SEQUENTIAL READER(VERSION IIA, RELEASE 2)  12/22/76     17.306
RUN-TIME PARAMETERS---->  NAME=PRES-SEQ  FIRST LAST                                    <---
            BLOCK DATA FILE WRITTEN
            DATA BASE TABLE FILE WRITTEN
 DATA BASE INITIALIZED WITH    50 PAGES.
THE TOTAL NUMBER OF RECORDS RETRIEVED IS--->     4290
```

Figure 7-24

from the sequential database. If there were errors during the run, the error messages will be printed after the SRIF initialization message.

The other output from the sequential Reader is a dump of a few records in the source RIF. Since this output is the same as in the IDS Reader, the user should refer to Section 7.5.5 and Figure 7-17 for an explanation and sample of that output.

8.0 RUNNING THE RESTRUCTURER

The Restructurer is the second of the three major modules of the Data Translator. After the Reader has produced an ADBMS database that is logically equivalent to the original IDS source database, the Restructurer must be run to perform the logical transformations specified by the TDL. The output is another ADBMS database in relational (rather than network) form. This database is then used by the Writer to produce a logically equivalent network IDS database.

The following checklist indicates the steps of the translation process which should be completed before attempting to run the Restructurer:

1. The user must generate SDDL tables for both the source and target databases.
2. The Reader must be run to produce a source RIF database.
3. A TDL description of the desired translation must be written and analyzed to generate TDL tables.
4. A random file must be created to hold the target RIF database. Its size must be a multiple of 12 llinks (see Section 8.4.1).

When the above steps have been completed, the user is ready to run the Restructurer.

8.1 Major Components of the Restructurer

Figure 8-1 shows the major components of the Restructurer. The arrows indicate the direction of information flow among the various components. The following is a brief explanation of each:

USER-INPUT DIRECTIVES: User-supplied statements that control various aspects of the Restructurer run (e.g., whether or not to initialize the target RIF database).

SOURCE RIF DATABASE: An ADBMS database, output by the Reader, which is logically equivalent to the user's original IDS database.

"NAMES" TDL TABLES: An ADBMS database, output by the TDL Analyzer, which contains all the translation information specified by the user's TDL description.

"POINTERS" TDL TABLES: A copy of the "NAMES" TDL tables in which all ADBMS names (e.g., record type names) have been replaced with symbolic pointers. The necessary processing is performed by the Restructurer prior to the actual restructuring of the database. This second copy of the TDL tables is used to avoid having to interpret the same ADBMS literals over and over again each time a target record is created or a source record accessed. This saves considerable processing time during the Restructurer run.

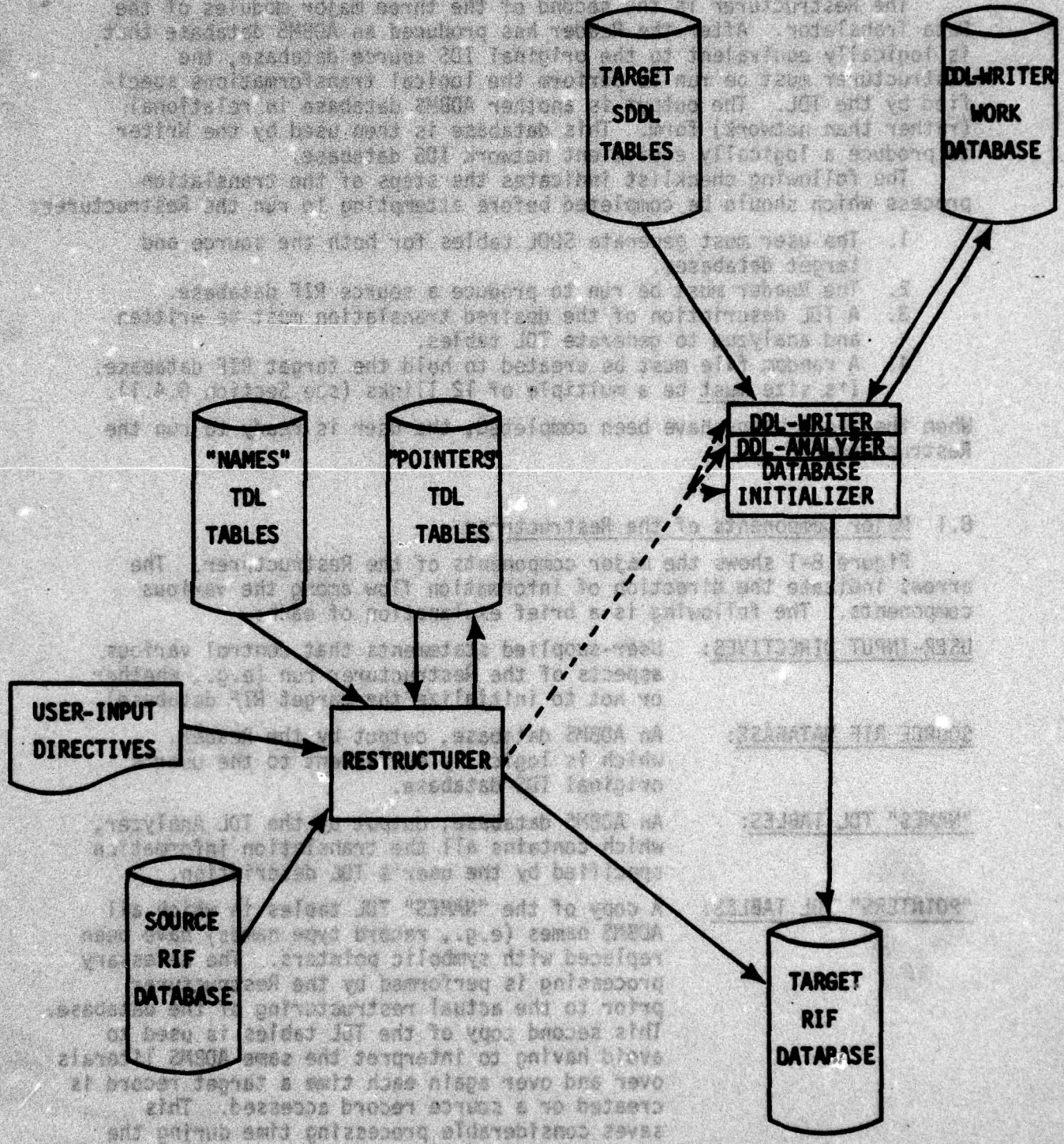


Figure 8-1

Restructurer - Major Components

RESTRUCTURER: Driven by the translation information specified in the TDL tables, the Restructurer builds the target RIF database from the data in the source RIF database.

TARGET SDDL TABLES: An ADBMS database, output by the IDS Analyzer, that describes the data structure of the target RIF database.

DDL-WRITER, DDL-ANALYZER, DATABASE-INITIALIZER: Three ADBMS support modules that are called by the Restructurer to (respectively) (1) read the data structure information from the target SDDL tables and produce ADBMS DDL text which describes the target RIF; (2) analyze this DDL text to produce an ADBMS DBTF; and (3) use this DBTF to initialize the target RIF database file so that records can be stored in it.

DDL-WRITER WORK DATABASE: An ADBMS scratch database used by the DDL Writer to produce ADBMS DDL text.

TARGET RIF DATABASE: An ADBMS database in which are stored the target record instances built by the Restructurer. This database is used as input to the Writer to produce the user's final IDS target database.

8.2 Restructurer Processing Flow

The restructuring process consists of two phases. In Phase I, general housekeeping functions are performed: all databases are opened; the User Input Directives are processed; if directed to do so, the target RIF database is initialized; and, the "POINTERS" copy of the TDL tables is prepared.

Phase II is the actual restructuring of the database. The Restructurer puts groups of "compatible" access paths on a "stack", and uses this "stack" to exhaustively access all source record instances (represented by the combined access paths) in a pseudo-pre-order tree fashion. For each group of source record instances accessed, the corresponding target record instances are constructed and stored in the target RIF database. When all access paths have been processed, the Restructurer is done, and a statistical summary is printed at the end of the Restructurer report.

8.3 Restructurer Control Card Deck Setup

Figure 8-2 shows the prototype control card deck setup for running the Restructurer. There are three activities: (1) make temporary copies of all databases, (2) run the Restructurer, and (3) copy the target RIF back to the permanent file.

```
0020    $       IDENT   YOUR-IDENT.LABEL
0030    $       NOTE
0040    $       NOTE    ************************************************************
0050    $       NOTE    *  ACTIVITY 01 -- ALL DATABASES TO BE USED BY THE        *
0060    $       NOTE    *  RESTRUCTURER ARE COPIED FIRST TO PROTECT THE          *
0070    $       NOTE    *  ORIGINALS.  BE SURE TO PUT THE CORRECT SIZES          *
0080    $       NOTE    *  (IN RANDOM LINKS) ON THE $FILE CARDS.                 *
0090    $       NOTE    ************************************************************
0100    $       UTILITY
0110    $       NOTE
0120    $       FUTIL   01.02.RCOPY/1F/
0130    $       PRMFL   01.R.R.***SOURCE RIF DATABASE***
0140    $       FILE    02.X0S.***SOURCE RIF DATABASE SIZE***
0150    $       NOTE
0160    $       FUTIL   03.04.RCOPY/1F/
0170    $       PRMFL   03.W.R.***TARGET RIF DATABASE***
0180    $       FILE    04.X1S.***TARGET RIF DATABASE SIZE***
0190    $       NOTE
0200    $       FUTIL   05.06.RCOPY/1F/
0210    $       PRMFL   05.R.R.***TDL TABLES DATABASE***
0220    $       FILE    06.X2S.***TDL TABLES DATABASE SIZE***
0230    $       NOTE
0240    $       FUTIL   07.08.RCOPY/1F/
0250    $       PRMFL   07.R.R.***TDL TABLES DATABASE***
0260    $       FILE    08.X3S.***TDL TABLES DATABASE SIZE***
0270    $       NOTE
0280    $       FUTIL   09.10.RCOPY/1F/
0290    $       PRMFL   09.R.R.***TARGET SDDL TABLES DATABASE***
0300    $       FILE    10.X4S.***TARGET SDDL TABLES DATABASE SIZE***
0310    $       NOTE
0320    $       FUTIL   11.12.RCOPY/1F/
0330    $       PRMFL   11.R.R.***DDL-WRITER WORK DATABASE***
0340    $       FILE    12.X5S.12R
0350    $       NOTE
0360    $       NOTE    ************************************************************
0370    $       NOTE    *  ACTIVITY 02 -- RUN THE RESTRUCTURER.  NOTE THAT       *
0380    $       NOTE    *  THE $LIMITS CARD MUST BE ADJUSTED FOR EACH            *
0390    $       NOTE    *  PARTICULAR RESTRUCTURING JOB.                         *
0400    $       NOTE    ************************************************************
0410    $       EXECUTE DUMP.NREST
0420    $       LIMITS  CPU-TIME.CORE.-5K.OUTPUT-LINE-LIMIT
0430    $       PRMFL   R*.R.S.***RESTRUCTURER R* FILE***
0440    $       FILE    H*..40R
0450    $       PRMFL   TR.R.R.***TRANSLATOR LIBRARY***
0460    $       NOTE
0470    $       NOTE    ------------------------------------------
0480    $       NOTE      THE FOLOWING SIX $FILE CARDS CORRESPOND
0490    $       NOTE      TO THE SIX DATABASES COPIED IN ACTIVITY 01
0500    $       NOTE
0510    $       FILE    10.X0R          'SOURCE RIF
0520    $       FILE    11.X1S          'TARGET RIF (SAVED FOR ACTIVITY 03)
0530    $       FILE    12.X2R          'NAMES' TDL TABLES
0540    $       FILE    13.X3R          'POINTERS' TDL TABLES
0550    $       FILE    14.X4R          TARGET SDDL TABLES
0560    $       FILE    15.X5R          DDL-WRITER WORK DATABASE
```

Figure 8-2

Prototype Restructurer Control Card Setup

```
0570  $     NOTE    ------------------------------------------
0580  $     NOTE      THE FOLLOWING FOUR $FILE CARDS ARE USED
05 0  $     NOTE      IN THE PREPARATION OF THE TARGET RIF FOR
0600  $     NOTE      RESTRUCTURING
0610  $     NOTE
06 0  $     FILE    01.NULL          OUTPUT SINK (DDLA. DBINT)
06 0  $     FILE    02               DDL TEXT FOR TARGET RIF
0640  $     FILE    03               DBTF FOR TARGET RIF
06 0  $     FILE    04               USER HASH INPUT FILE TO DBINT
06 0  $     NOTE    ------------------------------------------
0670  $     NOTE
0680  $     DATA    05..COPY
06 0  $     NOTE    ------------------------------------------
0700  $     NOTE      INCLUDED HERE SHOULD BE EITHER
0710  $     NOTE         1) USER-INPUT STATEMENTS. OR
07 0  $     NOTE         2) $SELECTA TO FILE CONTAINING USER-
07 0  $     NOTE          INPUT STATEMENTS
0740  $     NOTE    ------------------------------------------
07 0  $     ENDCOPY
07 0  $     NOTE
0770  $     NOTE    ************************************************************
07 0  $     NOTE    *  ACTIVITY 03 -- COPY THE NEW TARGET RIF DATABASE         *
07 0  $     NOTE    *  BACK TO THE PERMANENT FILE.  IF THE RESTRUCTURER        *
0800  $     NOTE    *  ENCOUNTERED AN ERROR IN ACTIVITY 02. THE $IF CARD       *
0810  $     NOTE    *  WILL PREVENT OVER-WRITING THE OLD TARGET DATABASE       *
08 0  $     NOTE    *  FILE WITH THE (POSSIBLY INCONSISTENT) NEW ONE.          *
08 0  $     NOTE    ************************************************************
0840  $     IF      35.ENDJOB
08 0  $     UTILITY
08 0  $     FUTIL   01.02.RCOPY/IF/
0870  $     FILE    01.XIR
0880  $     PRMFL   02.W.R.***TARGET RIF DATABASE***
08 0  $     ENDJOB
```

Figure 8-2 (cont'd).

LINE NUMBER	NOTES
0020	Standard $IDENT card
0130, 0170, 0210 0250, 0290, 0330	$PRMFL cards for all required databases
0420	$LIMITS card for Restructurer run
0430	$PRMFL for Restructurer R* file
0450	$PRMFL for Translator Library file
0690-0740	Must be replaced with User Input Directives
0880	Target RIF copied back to this $PRMFL file.

8.4 Activity 01 Control Cards

Figure 8-3 shows the control cards for Activity 01. This activity makes temporary copies of all databases to be used by the Restructurer in Activity 02.

8.4.1 File Code Assignments

Lines 0130-0140: The source RIF database (output of the Reader) should be assigned to file code 01 via the $PRMFL card on line 0130. The correct size parameter must be inserted in the $FILE card on line 0140. It must be at least as large as the random file containing the source RIF database, and may be larger. For example, if

(1) Source RIF in MICHIGAN/SOURCEDB, and
(2) SOURCEDB is 55 llinks,

then lines 0130-0140 should be

```
0130$  PRMFL  01, R, R, MICHIGAN/SOURCEDB
0140$  FILE   02, XOS, 5R
```

Note that 5 random links = 60 llinks, more than sufficient to hold SOURCEDB. SOURCEDB is copied to a temporary file (02) and saved for the next activity with LUD XO.

Lines 0170-0180: The target RIF database should be assigned to file code 03 via the $PRMFL card on line 0170. A corresponding size parameter should appear on line 0180. However, unlike the source RIF database, the target RIF database must be an integral multiple of 12 llinks, and the size parameter on the $FILE card must match exactly. The reason for this is that after the Restructurer finishes storing records in the target RIF, the temporary copy is copied back to the permanent file (Activity 03). Since temporary files are always multiples of 12 llinks, the target RIF must also be a multiple of 12 llinks so that a copy can be made

```
0030    $       NOTE
0040    $       NOTE    ************************************************************
0050    $       NOTE    *  ACTIVITY 01 -- ALL DATABASES TO BE USED BY THE          *
0060    $       NOTE    *  RESTRUCTURER ARE COPIED FIRST TO PROTECT THE            *
0070    $       NOTE    *  ORIGINALS.  BE SURE TO PUT THE CORRECT SIZES            *
0080    $       NOTE    *  (IN RANDOM LINKS) ON THE $FILE CARDS.                   *
0090    $       NOTE    ************************************************************
0100    $       UTILITY
0110    $       NOTE
0120    $       FUTIL   01,02,RCOPY/1F/
0130    $       PRMFL   01,R,R,***SOURCE RIF DATABASE***
0140    $       FILE    02,X0S,***SOURCE RIF DATABASE SIZE***
0150    $       NOTE
0160    $       FUTIL   03,04,RCOPY/1F/
0170    $       PRMFL   03,W,R,***TARGET RIF DATABASE***
0180    $       FILE    04,X1S,***TARGET RIF DATABASE SIZE***
0190    $       NOTE
0200    $       FUTIL   05,06,RCOPY/1F/
0210    $       PRMFL   05,R,R,***TDL TABLES DATABASE***
0220    $       FILE    06,X2S,***TDL TABLES DATABASE SIZE***
0230    $       NOTE
0240    $       FUTIL   07,08,RCOPY/1F/
0250    $       PRMFL   07,R,R,***TDL TABLES DATABASE***
0260    $       FILE    08,X3S,***TDL TABLES DATABASE SIZE***
0270    $       NOTE
0280    $       FUTIL   09,10,RCOPY/1F/
0290    $       PRMFL   09,R,R,***TARGET SDDL TABLES DATABASE***
0300    $       FILE    10,X4S,***TARGET SDDL TABLES DATABASE SIZE***
0310    $       NOTE
0320    $       FUTIL   11,12,RCOPY/1F/
0330    $       PRMFL   11,R,R,***DDL-WRITER WORK DATABASE***
0340    $       FILE    12,X5S,12R
```

Figure 8-3
Activity 01 Control Cards

from and later copied back to the same file. The target RIF is copied to a temporary file (04) and saved for the next activity with LUD X1.

Lines 0210-0220: The TDL tables database should be assigned to file code 05 via the $PRMFL card on line 0210. A corresponding size parameter should appear on line 0220. The size of the temporary file may be larger than the permanent file, as in the example for the source RIF database. The TDL tables are copied to a temporary file (06) and saved for the next activity with LUD X2. This copy will become the "NAMES" copy of the TDL tables in Activity 02 (see Section 8.1 for an explanation of "NAMES" vs. "POINTERS" TDL tables).

Lines 0250-0260: Same as lines 0210-0220. The same TDL tables database (file code 07) is copied to a temporary file (08) and saved for the next activity with LUD X3. This copy will become the "POINTERS" copy of the TDL tables in Activity 02.

Lines 0290-0300: The SDDL tables database for the target database should be assigned to file code 09 via the $PRMFL card on line 0290. A corresponding size parameter should appear on line 0300 (may be larger than permanent file size). The target SDDL tables are copied to a temporary file (10) and saved for the next activity with LUD X4.

Line 0330: The DDL Writer Work Database (restored from the SYSGEN tape) should be assigned to file code 11 via the $PRMFL card on line 0330. Its size is fixed at 12 random links = 144 llinks. The Work Database is copied to a temporary file (12) and saved for the next activity with LUD X5.

8.4.2 User Parameters

No user parameters are required for Activity 01.

8.4.3 Output Interpretation and Example

The output of Activity 01 is the standard $UTILITY output and is not shown here.

8.4.4 Errors

Any error messages will be standard control card or $UTILITY messages documented in H-6000 documentation. However, the following are some of the more elementary sources of errors:

1. Database file is sequential instead of random.
2. Size parameter on $FILE card is too small.
3. File name spelled incorrectly.

8.5 Activity 02 Control Cards

Figure 8-4 shows the control cards for Activity 02. This activity actually executes the Restructurer and produces a temporary copy of the target RIF database.

8.5.1 File Code Assignments

Line 0430: The Restructurer R* file (restored from the SYSGEN tape) should be assigned to file code R* via the $PRMFL card on line 0430. This file contains the Restructurer object code.

Line 0450: The Translator Library file (restored from the SYSGEN tape) should be assigned to file code TR via the $PRMFL card on line 0450. This library file contains all the support routines needed by the Restructurer.

8.5.2 User Parameters

There are three areas of user input to the Restructurer run: (1) the $ LIMITS card, (2) the User Input Directives on file code 05, and (3) optional use of $REMOTE and $PRMFL cards for file codes 01-04.

8.5.2.1 The $LIMITS Card

The $LIMITS card on line 0420 must be replaced with a suitable $LIMITS for each particular run. The following rules-of-thumb apply to runs made with #EXECUTION-MONITOR=OFF (see Section 8.5.2.2):

1. The CPU time required varies roughly linearly with the size of the database to be restructured. However, processor time requirements will be greater for runs with more complex access paths, more qualifications, and for runs that use user conversion and qualification routines. The general rule is: take the number of record instances in the larger of the two (source and target) databases. Divide this number by thirty (30). The result is the number that goes on the $LIMITS card for CPU-TIME. For example, suppose the source database contained 2000 record instances, and the target database was expected to have 3000 record instances. Then 3000÷30 = 100, and the $LIMITS card would look like:

   ```
   $  LIMITS  100, CORE,-5K,OUTPUT-LINE-LIMIT
   ```

   Note: this method provides a gross over-estimate so that a Restructurer run will not terminate due to lack of CPU time requested. After a few runs, the user should develop a feeling for how much time a particular run will take, and may cautiously lower his CPU limit if it is to his advantage.

```
0350    $       NOTE
0360    $       NOTE    ************************************************************
0370    $       NOTE    *  ACTIVITY 02 -- RUN THE RESTRUCTURER.  NOTE THAT      *
0380    $       NOTE    *  THE $LIMITS CARD MUST BE ADJUSTED FOR EACH           *
0390    $       NOTE    *  PARTICULAR RESTRUCTURING JOB.                        *
0400    $       NOTE    ************************************************************
0410    $       EXECUTE DUMP.NREST
0420    $       LIMITS  CPU-TIME.CORE.-5K.OUTPUT-LINE-LIMIT
0430    $       PRMFL   R*.R.S.***RESTRUCTURER R* FILE***
0440    $       FILE    H*..4OR
0450    $       PRMFL   TR.R.R.***TRANSLATOR LIBRARY***
0460    $       NOTE
0470    $       NOTE    ------------------------------------------
0480    $       NOTE      THE FOLOWING SIX $FILE CARDS CORRESPOND
0490    $       NOTE      TO THE SIX DATABASES COPIED IN ACTIVITY 01
0500    $       NOTE
0510    $       FILE    10.X0R          SOURCE RIF
0520    $       FILE    11.X1S          TARGET RIF (SAVED FOR ACTIVITY 03)
0530    $       FILE    12.X2R          'NAMES' TDL TABLES
0540    $       FILE    13.X3R          'POINTERS' TDL TABLES
0550    $       FILE    14.X4R          TARGET SDDL TABLES
0560    $       FILE    15.X5R          DDL-WRITER WORK DATABASE
0570    $       NOTE    ------------------------------------------
0580    $       NOTE      THE FOLLOWING FOUR $FILE CARDS ARE USED
0590    $       NOTE      IN THE PREPARATION OF THE TARGET RIF FOR
0600    $       NOTE      RESTRUCTURING
0610    $       NOTE
0620    $       FILE    01.NULL         OUTPUT SINK (DDLA. DBINT)
0630    $       FILE    02              DDL TEXT FOR TARGET RIF
0640    $       FILE    03              DBTF FOR TARGET RIF
0650    $       FILE    04              USER HASH INPUT FILE TO DBINT
0660    $       NOTE    ------------------------------------------
0670    $       NOTE
0680    $       DATA    05..COPY
0690    $       NOTE    ------------------------------------------
0700    $       NOTE      INCLUDED HERE SHOULD BE EITHER
0710    $       NOTE         1) USER-INPUT STATEMENTS. OR
0720    $       NOTE         2) $SELECTA TO FILE CONTAINING USER-
0730    $       NOTE            INPUT STATEMENTS
0740    $       NOTE    ------------------------------------------
0750    $       ENDCOPY
```

Figure 8-4

Activity 02 Control Cards

2. The CORE required will almost always be 66K, since the Restructurer buffers and storage are the same for each run. The only time more core is needed is if user qualification and/or conversion routines are used (see Section 4.2.3 on user qualification and conversion routines, and Section 8.7 for modifications to Restructurer control cards).

3. The default OUTPUT-LINE-LIMIT for a $EXECUTE activity (as is Activity 02) is more than sufficient if #EXECUTION-MONITOR=OFF.

If running with #EXECUTION-MONITOR=ON (see Section 8.5.2.2), the following adjustments should be made to the estimates presented above:

4. Increase the number obtained in (1) above for CPU-TIME by 25%.

5. The Execution-Monitor output is quite lengthy and is usually only needed when debugging a very stubborn bug in the user's TDL. If it is needed, only the access paths being debugged should be active for the run (see Section 8.5.2.2). The general rule is: take the number of record instances in the larger of the two databases. If only a few access paths are being used, this number may be scaled down by a factor equalling the fraction of the source database instances represented by the active access paths. In either case, multiply this number by fifty (50). This is the number that goes on the $LIMITS card for OUTPUT-LINE-LIMIT.

Presented here is a complete example of how to estimate the $LIMITS card. Assume the following:

a. Source database: 2000 record instances
b. Target database: 3000 record instances
c. No user qualification or conversion routines

Then, if #EXECUTION-MONITOR=OFF,

CPU-TIME = 3000 ÷ 30 = 100
CORE = 66K
OUTPUT-LINE-LIMIT not needed

And the $LIMITS card is

```
$   LIMITS   100,66K,-5K
```

If #EXECUTION-MONITOR=ON, then

CPU-TIME = 100 + (.25)100 = 125
OUTPUT-LINE-LIMIT = 3000 x 50 = 150,000

And the $LIMITS card is

```
$   LIMITS   125,66K,-5K,150000
```

8.5.2.2 User Input Directives

The User Input Directives are read by the Restructurer from file code 05, and are used to control various aspects of the Restructurer run. Lines 0690-0740 must be replaced (i.e., the $NOTE cards must not appear between lines 0680 and 0750 in final form) with either the actual User Input Directives

or with a $SELECTA to an ASCII file containing them. A description of the various directives is presented here, followed by an example.

1. All User Input Directives begin with a pound sign (#). However, since TSS strips off a pound sign immediately following a file line number, two pound signs should appear with the directive so that when the User Input Processor reads file code 05, each directive begins with one pound sign.

2. If a pound sign is desired as the first character of a TDLAP name, two pound signs should be used (##). Again, since TSS strips off one pound sign, three pound signs must follow the file line number to indicate that the TDLAP name begins with one pound sign. Pound signs within a name need no special treatment.

3. No embedded blanks are allowed in any User Input Directives.

4. User Input Directives must begin in column one (1). The first blank encountered on a line terminates the line. User Input Directives must be one-to-a-line and may not extend over more than one line.

5. The following are legal User Input Directives:

   a. #TRANSLATION-NAME=48-character-name

   This directive allows the user to put a label on the Restructurer output. The label may be up to 48 nonblank characters, and will appear in the User Input Summary and at the top of the Statistical Summary (see Section 8.5.3).

   b. #RUN = { INITIALIZE / CONTINUE }

   This directive determines whether or not the target RIF database is initialized before restructuring begins.

   INITIALIZE - the copy of the target RIF database file is initialized before restructuring. The contents of the permanent file are not modified until Activity 03.

   CONTINUE - the copy of the target RIF database file is not modified in any way. The target RIF is assumed to have been initialized (e.g., by a previous run) and may contain restructuring output (target records) from previous Restructurer runs. The output from the current run will simply be added to whatever already exists in the target RIF database.

   c. #EXECUTION-MONITOR={ ON / OFF }

   This directive turns the Restructurer debug output on or off:

   ON - debug output appears along with the Restructurer report on report code 06.

   OFF - no debug output is printed.

Note: error messages always appear on report code 06, independent of the #EXECUTION-MONITOR = directive.

d. #ACTIVE-TDLAPS

This directive indicates that the following TDLAPS are to be "active", i.e., used to build target records during the Restructurer run. TDLAP names must appear one per line immediately following the directive and must start in column one (1). The appearance of the keyword ALL-TDLAPS-ACTIVE instead of a TDLAP name indicates all TDLAPS are to be used for restructuring during the run. Other rules:

1. If ALL-TDLAPS-ACTIVE is used, it may not be followed by a TDLAP name.
2. Either ALL-TDLAPS-ACTIVE or at least one TDLAP name must follow the #ACTIVE-TDLAPS directive.
3. The list of TDLAP names is terminated by an end-of-file or another User Input Directive.
4. All occurrences of a particular TDLAP name after the first occurrence are ignored and do not produce an error message.

This directive is particularly useful for debugging one access path at a time, and for performing large restructuring operations a little at a time. For example, a particular restructuring job may take longer than the amount of continuous computer "up" time available. In cases like this, the user may make the first Restructurer run with #RUN=INITIALIZE and one or two access paths "active". Each succeeding run is then made with #RUN=CONTINUE and one or two different access paths "active" until all access paths have been processed. Thus, the first run initializes the target RIF database and stores the target records built from the "active" access paths. The target RIF database is not initialized on succeeding runs; the target records from each active access path are stored in the same database as in the first run until all restructuring is completed. The content of a target RIF database produced in this way is logically equivalent to a target RIF produced in a single run. Note, however, that if the source RIF database, source SDDL tables, and/or target SDDL tables are regenerated part way through this restructuring-by-parts process, previous runs may become inconsistent and will have to be rerun from scratch.

e. #USER-HASH-INPUT

This directive indicates that the following lines should be input to the ADBMS Database-Initializer. They will be used to modify the standard ADBMS hashing algorithm that is used to store records in the target RIF database. Use of this directive is necessary only in unusual cases when, for a particular target RIF database, the standard record storage algorithm breaks down and a different algorithm is required. This event necessitates a thorough analysis of the data before attempting to alter the record storage algorithm. Data Translation Project personnel should be consulted in the event that problems of this nature arise.

f. #END-USER-INPUT

This directive signals the end of the User Input Directives, and scanning of the directives terminates when this directive is encountered. If an end-of-file is encountered before this directive, it is assumed that the end of the User Input Directives has been reached.

Further Notes and Rules

6. The User Input Directives are order-independent.

7. Any errors detected while processing the User Input Directives will result in the cancellation of the Restructurer run. However, all directives are always processed for errors.

8. Any directive appearing more than once is considered an error.

9. Certain directives must appear for each run:

   a. #RUN=
   b. #EXECUTION-MONITOR=
   c. #ACTIVE-TDLAPS, followed by at least one TDLAP name or by ALL-TDLAPS-ACTIVE.

   If any of the above directives is missing, it is considered an error.

10. Optional directives are:

   a. #TRANSLATION-NAME=
   b. #USER-HASH-INPUT
   c. #END-USER-INPUT

   The absence of one or more of these is not an error.

11. Following are two examples of User Input Directives:

EXAMPLE #1

File Line No.	Directive
10	##TRANSLATION-NAME=OLD-TO-NEW
20	##RUN=INITIALIZE
30	##EXECUTION-MONITOR=OFF
40	##ACTIVE-TDLAPS
50	ALL-TDLAPS-ACTIVE
60	##END-USER-INPUT

Comments on Example #1

Line 10: Label for output
Line 20: Target RIF database is to be initialized
Line 30: No debug output is to be printed
Line 40: The following TDLAPS are to be used during the run:
Line 50: All of them
Line 60: End of User Input Directives

EXAMPLE #2

File Line No.	Directive
10	##RUN=CONTINUE
20	##EXECUTION-MONITOR=ON
30	##ACTIVE-TDLAPS
40	PEOPLE-REC
50	###OFFSPRING
(60)	end-of-file

Comments on Example #2

Line 10: Target RIF database not initialized; records are stored in database along with content from previous runs.
Line 20: Debug output will be printed on report code 06.
Line 30: The following TDLAPS will be "active" for the run.
Lines 40,50: The TDLAPS "PEOPLE-REC" and "#OFFSPRING" will be active.
Line (60): Nonexistent; end-of-file indicates end of User Input Directives

Note: Unlike the examples above, no intervening blanks should appear between the file line numbers and the User Input Directives.

8.5.2.3 Optional $REMOTEs and $PRMFLs

If desired, file codes 01, 02, 03, and 04 may be assigned to permanent files via $PRMFL cards (with WRITE permission) for later examination. In addition, line 0620 may be replaced by:

```
0620   $   REMOTE   01
```

which will force (usually unnecessary) output from the DDL Analyzer and Database Initializer to appear on report code 01. This output may be useful if errors occur in one or both of these modules.

8.5.3 Output Interpretation and Example

The *Restructurer Report* appears on report code 06 of Activity 02. (This description is for #EXECUTION-MONITOR=OFF). At the top of the first page will appear

```
######RESTRUCTURER REPORT--VERSION IIA, RELEASE 2#######
```

Any errors that occur between this message and the next will appear on this page. However, if all goes well, the remainder of the first page should be blank. At the top of the next page will be the message:

```
PHASE I - PROCESS USER INPUT FILE
```

User Input Processor error messages, if any, will follow. However, if no errors are detected, the remainder of the page will be blank.

At the top of the next page (following any error messages) will be the User Input Summary. The summary gives the status of each of the User Input Directives and information regarding the preparation of the target RIF database (if any occurred). An example of the User Input Summary is shown in Figure 8-5. If all User Input was correct, the last message of the summary will be "USER INPUT ACCEPTABLE."

The top of the next page should be headed by the message:

```
PHASE II - RESTRUCTURER BEGINS
```

with the remainder of the page blank (unless there are error messages). Each succeeding page will (again, if there are no errors) show a small summary of each stack built by the Stack Builder. An example is shown in Figure 8-6. The stack number simply indicates the order in which the stacks were built and accessed. A list of the access paths that were stacked together on that stack follows. Any error messages resulting from the accessing of the records on the stack will follow this list, ending with a message indicating whether or not the stack was successfully accessed.

```
USER INPUT SUMMARY
------------------

OPTION                  STATUS
------                  ------
TRANSLATION-NAME        NEW-PRESIDENTIAL-TO-DECADE
RUN                     INITIALIZE
EXECUTION-MONITOR       OFF
ACTIVE-TDLAPS           ALL-TDLAPS-ACTIVE
USER-HASH-INPUT         NONE SPECIFIED

  0 BAD CARDS DETECTED

NO ERRORS DETECTED IN USER INPUT
######RRIF DDL SUCCESSFULLY WRITTEN######

          DATA BASE TABLE FILE WRITTEN

 DATA BASE INITIALIZED WITH     6 PAGES.

 HASH DATA BASE HAS      5 HASH PAGES AND   0 OVERFLOW PAGES

 THERE ARE  1 BUCKETS/PAGE

HASH IS REMAINDER OF ((PK REDUCED  TO 1 WORD BY EXCLUSIVE OR)/      5)

USER INPUT ACCEPTABLE
```

Figure 8-5
Example Output - User Input Summary

```
THE FOLLOWING ACCESS PATHS HAVE BEEN SUCCESSFULLY STACKED ON STACK NO.   1
<PRES          >
<MARR          >
<OCCUPATION    >
<RELA          >

SOURCE INSTANCES FOR STACK NO.    1 EXHAUSTED: NO EPRORS DETECTED
```

Figure 8-6
Stack Summary

If there were no errors in the run, the page following the summary for the last stack will be headed by the message:

PHASE II COMPLETE - RESTRUCTURER TERMINATING

If errors occurred during the run, this message will be replaced by an appropriate error message. On this and the following pages appear the Statistical Summary. The source database summary follows the above message; the target database summary begins at the top of the page following the source database summary; and the access paths summary similarly follows the target database summary. Figures 8-7, 8-8, and 8-9 respectively show examples.

Following are explanations of Figures 8-7, 8-8, and 8-9.

Figure 8-7: Source Database

The #TRANSLATION-NAME appears under "STATISTICAL SUMMARY." (If none was specified via the use of the directive, the space will be blank). Information regarding each source record type is read across the columns from left to right.

"SOURCE RECORD TYPE" column: the six-character ADBMS name (as it appears in the source SDDL tables database) of each source record type that was accessed at least once (i.e., at least one attempt to retrieve an instance, actual or null, was successful).

"ACCESSES FOR THIS TYPE" column: the number of actual record instances accessed (retrieved) for each source record type. This number does not include null instances.

"PASSED QUALIFICATION" column: for each source record type, the number of record instances that satisfied qualification criteria. This number includes null instances.

"FAILED QUALIFICATION" column: for each source record type, the number of record instances that did not satisfy qualification criteria. This number also includes null instances.

The TOTALS at the bottom of each column are simply the sums of each column. Note that in general, as in Figure 8-7, the sum of columns 3 and 4 do not add up to give the corresponding number in column 2. The reason for this is twofold: (1) column 1 counts actual record instances retrieved, not including null instances, while columns 3 and 4 do include null instances, (2) when two or more compatible access paths "share" a record on the stack, the record need be retrieved only once; then all qualifications, one for each access path, may be performed on the same record without the need to access it again.

Figure 8-8: Target Database

Always starts at the top of a new page. The structure is similar to that of the source database summary.

"TARGET RECORD TYPE" column: the six-character ADBMS name (as it appears in the target SDDL tables database) of each target record type for which at least one instance was created and stored in the target RIF.

"TOTAL RECORDS CONSTRUCTED" column: the total number of record instances of each record type that the Restructurer attempted to create. This includes duplicate records.

PHASE II COMPLETE - RESTRUCTURER TERMINATING

STATISTICAL SUMMARY

NEW-PRESIDENTIAL-TO-DECADE

SOURCE DATABASE

SOURCE RECORD TYPE	ACCESSES FOR THIS TYPE	PASSED QUALIFICATION	FAILED QUALIFICATION
PRESID	147	448	0
STATEN	85	92	93
ADMINI	287	126	723
MARRIA	7	7	0
OCCUPA	14	14	0
RELATO	124	320	0
CONGRE	116	296	0
ELECTI	45	14	76
LOSERS	17	17	0
HOUSE	52	52	0
SENATE	52	52	0
VICE-P	7	7	0
CITIES	59	59	4
STATES	6	6	13
TOTALS	1018	1510	909

Figure 8-7

Source Database Statistical Summary

TARGET DATABASE
****

TARGET RECORD TYPE	TOTAL RECORDS CONSTRUCTED	DUPLICATE RECORDS CONSTRUCTED	NET RECORDS CONSTRUCTED
RELATO	26	12	14
OCCUPA	14	6	8
MARRIA	7	3	4
PRESID	7	3	4
LOSERS	17	0	17
ELECTI	7	0	7
SENATE	52	24	28
HOUSE	52	24	28
CONGRE	26	12	14
VICE-P	7	0	7
ADMINI	7	0	7
CITIES	59	0	59
STATEN	19	0	19
STATES	6	0	6
TOTALS	306	84	222

Figure 8-8

Target Database Statistical Summary

"DUPLICATE RECORDS CONSTRUCTED" column: the number of duplicate (i.e., had same primary key field values) record instances of each record type that the Restructurer attempted to create. These records are never stored in the target RIF database since a duplicate instance has already been stored.

"NET RECORDS CONSTRUCTED" column: the total number of record instances of each record type that was stored in the target RIF database.

The TOTALS are again the sums of each column. However, for the target database summary, it should always be true for each row (i.e., target record type) that

NET (col.4) = TOTAL (col. 2) - DUPLICATE (col.3)

Figure 8-9: Access Paths

Gives the following information:

"STACK NUMBER" column: these numbers correspond to the stack numbers given with each short stack summary earlier in the report.

"ACCESSED SUCCESSFULLY" column: tells whether or not the corresponding stack was processed with no errors:

YES = no errors
NO = errors occurred; results from this stack are probably not consistent.
XXX = a job abort occurred while processing this stack; results are unpredictable

"SOURCE ACCESSOR TIMES IN MILLISECONDS" columns: gives timing data for each stack in milliseconds (1 second = 1000 milliseconds):

"ELAPSED TIME": the elapsed time for the accessing of the stack.
"PROC. TIME": the CPU (processor) time used for the accessing of the stack.

For example, note that stack number 1 took an elapsed time of 86.398 seconds to process while using up 22.137 seconds of processor time. These times give an indication of the relative complexities of and the number of source record instances corresponding to the various access paths.

"ACCESS PATH INSTANCES FROM STACK" column: for each stack, the number of access path instances from all the access paths on the stack. Each access path instance corresponds to one record instance creation by the Restructurer. The TOTAL for this column should equal the TOTAL for column 2, "TOTAL RECORDS CONSTRUCTED", from the target database summary.

"ACCESS PATHS ON STACK" column: a list of the access paths on the corresponding stack. For example, in Figure 8-9, the access paths named PRES, MARR, OCCUPATION, and RELA were all processed together (because they were compatible) on stack number 1.

"ACCESS PATH INSTANCES FROM A.P." column: same as "ACCESS PATH INSTANCES FROM STACK", except for each access path.

The TOTALS are again the sum of the corresponding columns. The final message should be the "DONE" message following the Access Path Summary, as shown in Figure 8-9.

ACCESS PATHS
****

STACK NUMBER	ACCESSED SUCCESSFULLY	SOURCE ACCESSOR TIMES IN MILLISECONDS ELAPSED TIME	PROC. TIME	ACCESS PATH INSTANCES FROM STACK	ACCESS PATHS ON STACK	ACCESS PATH INSTANCES FROM A.P.
1	YES	86398	22137	54	PRES	7
					MARR	7
					OCCUPATION	14
					RELA	26
2	YES	30534	9132	24	ELEC	7
					LOSE	17
3	YES	177330	55824	130	CONG	26
					HOUS	52
					SENA	52
4	YES	23691	9454	14	ADMI	7
					VICE	7
5	YES	114895	22379	84	STAT	19
					CITI	59
					STAU	6
	TOTALS	432848	118926	306		306

######RESTRUCTURER DONE######

Figure 8-9

Access Paths Statistical Summary

Some Final Notes on the Restructurer Report

1. An error-free run should be mostly empty space; the numerous page skips make the report easier to analyze in the case of numerous errors.

2. The Statistical Summary may be inconsistent for runs that were aborted or in which errors occurred.

8.5.4 Errors

The Restructurer is designed to continue executing in spite of errors as long as they are not of a nature that would make further execution useless (e.g., unable to open a database). Extensive error messages are provided and printed out on report code 06 along with the Restructurer report. Appendix D contains an explanation of all Restructurer error messages and suggests possible sources of most errors.

Most Restructurer error messages begin with six asterisks (******). #EXECUTION-MONITOR debug messages always begin with six pound signs (######).

8.6 Activity 03 Control Cards

Figure 8-10 shows the control cards for Activity 03. If the Restructurer run (Activity 02) was successful, the temporary copy of the target RIF database is copied back to the permanent file. If the run had errors in it, the $IF card (line 0840) will cancel Activity 03 to avoid destroying the old contents of the permanent file with a bad database.

8.6.1 File Code Assignments

Line 0880: the target RIF database should be assigned to file code 02 via the $PRMFL card on line 0880. No size parameter is needed. This should be the same permanent file as that assigned to file code 03 in Activity 01.

8.6.2 User Parameters

No user parameters are required for Activity 03.

8.6.3 Output Interpretation and Example

The output of Activity 03 is the standard $UTILITY output and is not shown here.

8.6.4 Errors

Any error messages will be standard control card or $UTILITY errors, documented in H-6000 documentation. Common errors are mentioned in Section 8.4.4.

```
0760    $       NOTE
0770    $       NOTE    ************************************************************
0780    $       NOTE    *  ACTIVITY 03 -- COPY THE NEW TARGET RIF DATABASE         *
0790    $       NOTE    *  BACK TO THE PERMANENT FILE.  IF THE RESTRUCTURER        *
0800    $       NOTE    *  ENCOUNTERED AN ERROR IN ACTIVITY 02, THE SIF CARD       *
0810    $       NOTE    *  WILL PREVENT OVER-WRITING THE OLD TARGET DATABASE       *
0820    $       NOTE    *  FILE WITH THE (POSSIBLY INCONSISTENT) NEW ONE.          *
0830    $       NOTE    ************************************************************
0840    $       IF      35.ENDJOB
0850    $       UTILITY
0860    $       FUTIL   01.02.RCOPY/IF/
0870    $       FILE    01.XIR
0880    $       PRMFL   02.W.R.***TARGET RIF DATABASE***
0890    $       ENDJOB
```

Figure 8-10

Activity 03 Control Cards

8.7 Modifications to Restructurer Control Cards for User Routines

The control card setup for running the Restructurer that was presented in Sections 8.3-8.6 must be modified if user qualification and/or conversion routines were specified in the TDL description (see Section 4.2.3). The object code for each user routine must be inserted into the Restructurer R* file along with the normal Restructurer object code. The modified R* file is then passed to Activity 02, where dynamic linking and loading are used to interface with the user routines.

Figure 8-11 shows the extra control cards required to run the Restructurer with user routines. This set of control cards is inserted between lines 0020 and 0030 of Figure 8-2; it then becomes Activity 01. Activity 02 is now the $UTILITY copy of all databases; Activity 03, the Restructurer; and Activity 04 copies the target RIF database back to the permanent file.

Line 0030: the Restructurer R* file (restored from the SYSGEN tape) must be assigned to file code *R via the $PRMFL card on line 0030.

Line 0060: this card copies the Restructurer object code from file code *R to file code R* up to the subroutine TDBSTA, which is the last subroutine in the Restructurer R* file. The object code for the user routines is inserted into the modified R* following this subroutine. This card should not be modified in any way.

Line 0140: "USER-NAME" must be replaced by the six-character name that appeared in a "WHEN QUALIFIED BY" or "CONVERT WITH" clause in the user's TDL description. Remember that this name cannot be the same as the subroutine name.

Line 0150: "SUBNAM" must be replaced by the subroutine name of the user routine.

Line 0160: "OBJECT-FILE" must be replaced by the file in which the user routine object code resides.

Lines 0140-0160 are repeated for each user routine. Line 0430 of Figure 8-2 must be replaced by

```
$     FILE     R*,RØR
```

The CORE option on the $LIMITS card (line 0420 of Figure 8-2) should also be increased from 66K according to the size of user routines. Finally, all the $NOTE cards in Figure 8-11 (lines 0080-0130 and 0170-0180) should be deleted before inserting it into Figure 8-2 (see example below).

As an example, assume the following clauses appear in the TDL description:

```
...WHEN QUALIFIED BY UQUAL1
          .
          .
          .
...CONVERT WITH UCONV1
          .
          .
          .
...CONVERT WITH UCONV2
```

Assume also that the user has written the following subroutines and compiled them into the corresponding files:

TDL NAME	SUBROUTINE	OBJECT-CODE FILE
UQUAL1	UUQUA1	MICHIGAN/UUQUA1.0
UCONV1	UUCON1	MICHIGAN/UUCON1.0
UCONV2	UUCON2	MICHIGAN/UUCON2.0

Figure 8-12 shows the control cards that must be inserted between lines 0020 and 0030 of Figure 8-2. Recall also that line 0430 of Figure 8-2 must be altered along with the $LIMITS card (line 0420), as specified above.

A final note on using user routines: it is suggested that the user adopt a naming convention such as beginning all "WHEN QUALIFIED BY," "CONVERT WITH," and user routine subroutine names with a double consonant (e.g., "ZZUSER"). This will avoid control card and loader errors caused by a user routine or $LINK name having the same name as a Restructurer subroutine or $LINK name.

```
0020    $       FILEDIT
0030    $       PRMFL   *R,R,S,***RESTRUCTURER R* FILE***
0040    $       FILE    R*,ROS,10L
0050    $       DATA    *C,,COPY
0060    $       COPY    ,,TDBSTA        LAST SUBR. IN RESTR. R*
0070    $       INCLUDE
0080    $       NOTE
0090    $       NOTE    ----------------------------------------
0100    $       NOTE      THE NEXT THREE CONTROL CARDS MUST BE
0110    $       NOTE      REPEATED AS A BLOCK FOR EACH USER
0120    $       NOTE      ROUTINE TO BE INCLUDED FOR THE RUN.
0130    $       NOTE
0140    $       LINK    USER-NAME       NAME FROM TDL DESCRIPTION
0150    $       ENTRY   SUBNAM          CONTROL TRANSFERRED TO THIS SUBR.
0160    $       SELECTD OBJECT-FILE     OBJECT CODE FOR USER ROUTINE
0170    $       NOTE    ----------------------------------------
0180    $       NOTE
0190    $       ENDEDIT
0200    $       ENDCOPY
```

Figure 8-11

Restructurer Control Cards Required
When Running with User Routines

```
0020    $       FILEDIT
0030    $       PRMFL   *R,R,S,MICHIGAN/RESTR.RS
0040    $       FILE    R*,ROS,10L
0050    $       DATA    *C,,COPY
0060    $       COPY    ,,TDBSTA        LAST SUBR. IN RESTR. R*
0070    $       INCLUDE
0140    $       LINK    UQUAL1
0150    $       ENTRY   UUQUA1
0160    $       SELECTD MICHIGAN/UUQUA1.O
0140    $       LINK    UCONV1
0150    $       ENTRY   UUCON1
0160    $       SELECTD MICHIGAN/UUCON1.O
0140    $       LINK    UCONV2
0150    $       ENTRY   UUCON2
0160    $       SELECTD MICHIGAN/UUCON2.O
0190    $       ENDEDIT
0200    $       ENDCOPY
```

Figure 8-12

Example Control Cards to Insert User Routines into Restructurer R*

9.0 RUNNING THE WRITER

As the final step in data translation it is necessary to transfer the record instances from the target database produced by the Restructurer (the target RIF) which is in a relational form supported by ADBMS to the user's target IDS database(s). This process is known as writing and is performed by the Writer. The Writer's sole duty is to preserve within the new target IDS database(s) all record and relation instances represented in the target RIF. As noted previously in the User Manual, the Writer will produce only target IDS databases (not ISP or sequential files). Up to five different IDS databases can be written from a single target RIF file.

For very large target databases, the Writer has a useful feature which permits partial database writes. This allows the user to split the execution of the Writer into different machine-time runs in case computer resources are restricted. For example, assume a database comprises record types A-Z, each of which consists of 2,000 instances. Further assume that no more than two hours of CPU time are available in any one block. Since it is not possible to write out 52,000 records in less than two hours, the partial database write feature can be employed to permit the user to store record types A-F on one run, to process record types G-R the next day, and finally, at a later date, finish the writing with record types S-Z. This feature is described in complete detail in Section 9.7.

Before executing the Writer it is necessary that all Data Translator steps have been successfully completed. These steps are summarized below.

1. Write source and target IDS MD sections.
2. Write and analyze the source and target extended (with level 61 entries) IDS MD sections. Use the IDS Analyzer yielding source and target SDDL tables.
3. Ensure that the Reader produces the source RIF database from up to five source IDS, ISP or sequential files (or combination thereof).
4. Write and analyze Translation Definition Language statements by using the TDL Analyzer to produce TDL tables.
5. Execute the Restructurer to output a target RIF which contains up to five target IDS database records.

The user additionally must create via FILSYS, IDS database files (subfiles, if desired) with the desired IDS attributes (PAGESIZE, INVENTORY, etc.) for each of the target IDS databases represented in the target RIF. Care must be taken to ensure that the database size is large enough or the Writer will have to be rerun.

WARNING: It is absolutely imperative that the description of the physical layout of target IDS records in the target MD section and the extended target MD section match each other perfectly. Specifically:

1. All items in the IDS MD and extended IDS MD sections must have the same relative order and length.
2. The 98 level entries must appear in exactly the same order in both target IDS MD and extended IDS MD sections.

3. The record type values (e.g. TYPE IS XXX) must be identical for each record in both target IDS MD and extended IDS MD sections.
4. The class (e.g. CALC, PRIMARY, SECONDARY) of a record must be the same in both target IDS MD and extended IDS MD sections.

9.1 Detailed Overview of Writer Execution

Executing the Writer requires five activities (with activity 2 being deleted if the second through nth run of a partial database write is being done). These are denoted as follows.

Activity	
1	UTILITY copy of target RIF database into a temporary file.
2	IDS QUTI execution on the target IDS database being written. Note that this activity cannot be performed if partial database writing is in progress. If three executions of the Writer are required to produce one target IDS database, then QUTI should be called only for the first run, not the second or third, or the page header records will be overwritten.
3	Compile the target IDS MD section into the COBOL portion of the Writer. Because the Writer uses IDS routines to store and link records, object-time version of the target IDS MD section must be available at execution time. Hence, the user is supplied with the source code for the COBOL portion of the Writer, in which the user must insert the target IDS MD section, compile, and then execute in activity four. See Section 9.6 for complete details.
4	Execute the Writer. The compiled COBOL program from activity three calls in the Writer routines supplied on an R* file. User parameters are supplied to specify which database and which records within that database are to be written. A permanent record is made of which records were written if a partial database write is in progress.
5	If the Writer executed with no fatal errors, the temporary copy of the target RIF (modified during activity four) is recopied back into the original target RIF file. Subsequent partial database writes require that the target RIF file (modified for any given run i by activity four) be the target RIF for the next run $i + 1$.

The Writer execution is illustrated in Figure 9-1. Each component of the job is summarized below.

Target RIF database - Output PRMFL of the Restructurer. It is an ADBMS database containing the representations of up to five target user databases. All record instances of a given type are linked together. Relation (set) instances are maintained in a relational way by

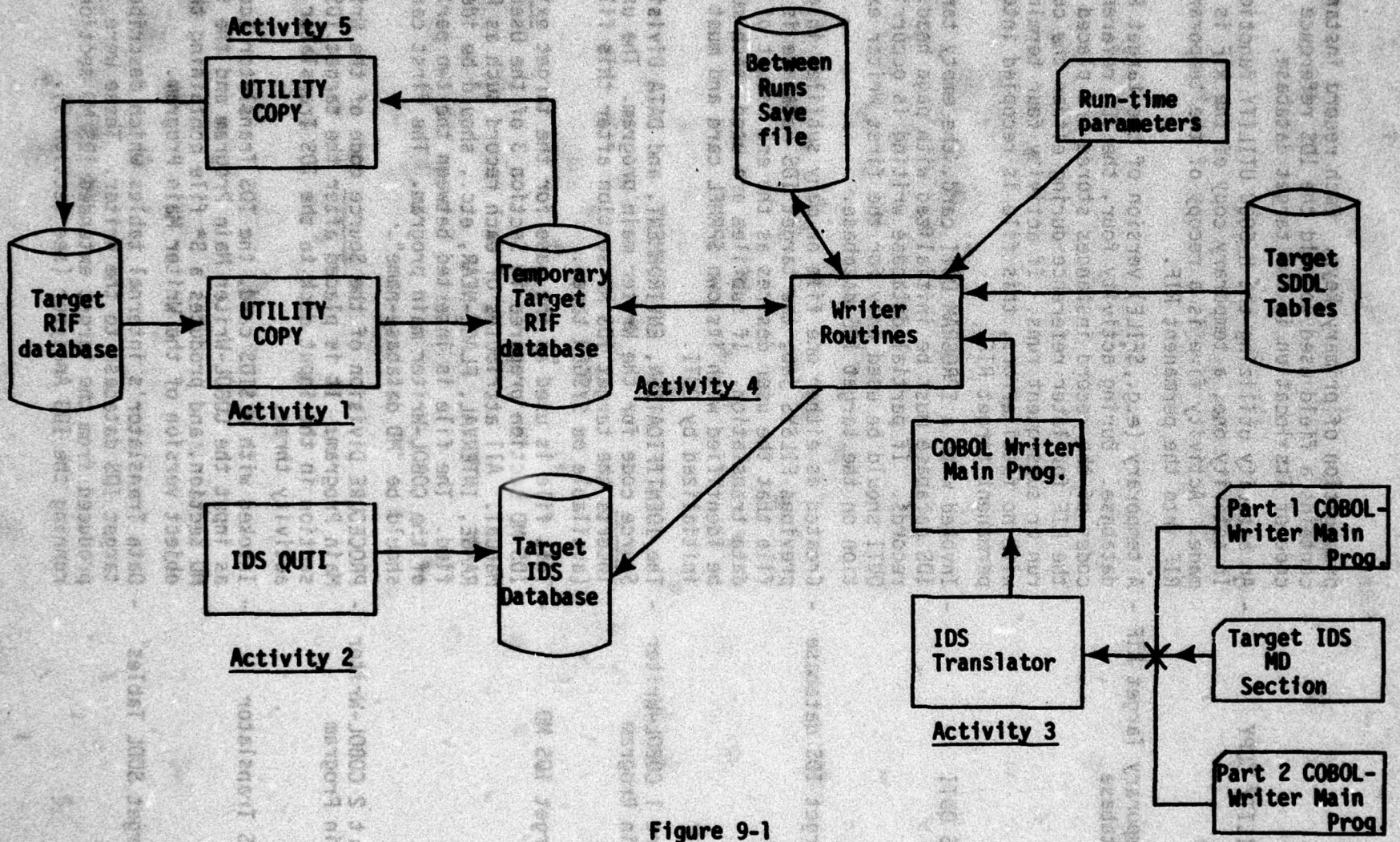


Figure 9-1
Executing the Writer

propagation of primary keys. Each record instance contains a field used to hold the IDS reference code of its location in the target database.

UTILITY COPY - An activity utilizing the H-6000 UTILITY function. In activity one, a temporary copy of the RIF is made. Activity five is a recopy of the temporary RIF into the permanent RIF.

Temporary Target RIF database - A temporary (e.g., $FILE) version of the target RIF database. During activity four, the IDS reference codes of the record instances stored are placed into the RIF for later reference during either the current run or subsequent runs. If activity four terminates with no fatal errors, this file is recopied into the permanent target RIF.

IDS QUTI - Invoked with a $PROGRAM QUTI card, the empty target IDS database must be initialized with page header records. If partial database writing is occurring, QUTI should be used only for the first Writer execution on the target IDS database.

Target IDS database - Created as either one file or many subfiles in a previous FILSYS job, the target IDS database is the file that the user desires as the end result of all data translation. If subfiles are used, each must be identified with its own $PRMFL card and must be initialized by QUTI.

Part 1 COBOL-Writer Main Program - The IDENTIFICATION, ENVIRONMENT, and DATA Division Source code for the Writer main program. The user inserts the target IDS MD section after this file (available on SYSGEN tape).

Target IDS MD - This file is used as the base for the target extended IDS MD section prepared in Section 3 of the User Manual. All attributes of each record such as PAGE-RANGE, INTERVAL, PLACE-NEAR, etc., should be identified. The file is inserted between the two parts of the COBOL-Writer main program. The first card should be "MD database-name".

Part 2 COBOL-Writer Main Program - PROCEDURE Division of the Source code of the Writer Main Program. It is placed after the target IDS MD section in the input deck to the IDS Translator for activity three.

IDS Translator - Invoked with a $IDS card, the IDS Translator accepts as input the COBOL-Writer Main Program and the target MD section, and produces a B* file containing the object version of the Writer Main Program.

Target SDDL Tables - Data Translator's internal tables which describe the target IDS database to the Writer. These were produced from the target extended IDS MD section by running the IDS Analyzer (see Section 5).

Run-time parameters - Specification to the Writer to indicate which of the databases (of five possible) in the target RIF is to be written into an IDS database. Directives concerning total or partial database writing are supplied.

Between runs save file - If partial database writes are being used, it is necessary to store information concerning which records were written on each partial execution. This information is maintained in a file which is sequential (two links is sufficient). There must be one "between runs save file" per target database.

Writer routines - The Writer R* file (available on the SYSGEN tape).

9.2 Explanation of Processing Flow

This section sketches a brief overview of the algorithms, inputs and outputs of the entire Writer execution. Note that at minimum, the Writer must be executed once per target database to be written. Additionally, each database write can be separated into partial database writes if computer resources are limited.

9.2.1 Activity 1 - Utility Copy of Target RIF

Inputs

1. PRMFL for target RIF database

Outputs

1. FILE for RIF database

Algorithm

A $FUTIL filecode, filecode, RCOPY/1F/ directive is supplied to UTILITY.

9.2.2 Activity 2 - Initialize IDS database

Inputs

1. An empty IDS database represented by either
   a) 1 $PRMFL
   b) multiple $PRMFLs if subfiles are used
2. Directives to initialize the IDS database

Outputs

1. An initialized IDS database

Algorithm

Each subfile must have its own IDS INITIAL directive for its page-range. Note that this activity should not be included if the Writer execution is not the first run on a given database file. That is, if n partial writes are required to produce one target IDS database, QUTI should be used only for the first run.

9.2.3 Activity 3 - Compile COBOL-Writer Main Program

Inputs

1. Source code for part 1 of the Writer Main program
2. User's target IDS MD section
3. Source code for part 2 of the Writer Main program

Outputs

1. A B* file containing an object COBOL-IDS program with the target IDS database represented by the common area .IDS..

Algorithm

A simple COBOL-IDS compile activity is performed. No permanent object file is required.

9.2.4 Activity 4 - Execute the Writer

Inputs

1. Run-time parameter file
2. Target SDDL tables
3. Between runs save file
4. Target RIF file (temporary copy)

Outputs

1. Execution report
2. Error report
3. Target IDS database
4. Updated Target RIF file
5. Updated between runs save file

Algorithm

Activity 4 can be viewed in the following two ways.
a) Multiple and partial database writing.
b) Process of transferring records from target RIF to the target IDS file.

A. Multiple and Partial Database Writing

The control over which database to write is determined by which values are placed on the run-time parameter cards. The user must specify the following:

1. Database name
2. First run for this database? (YES or NO)
3. All or some of the records for the database being written (ALL or PART)
4. Names of the records to be written for this partial database write (only if "PART" specified).

The user must supply to the Writer a permanent file in which notation of the record types written for each partial database write is made. A separate linked file is necessary for each target database. This file is known as the between runs save file. It need not be supplied if a total database write is in progress. However, it must be supplied for every partial write execution if partial writes are used. The flowchart below summarizes the steps that the user must take. Section 9.7 has the complete syntax rules for the runtime parameter file.

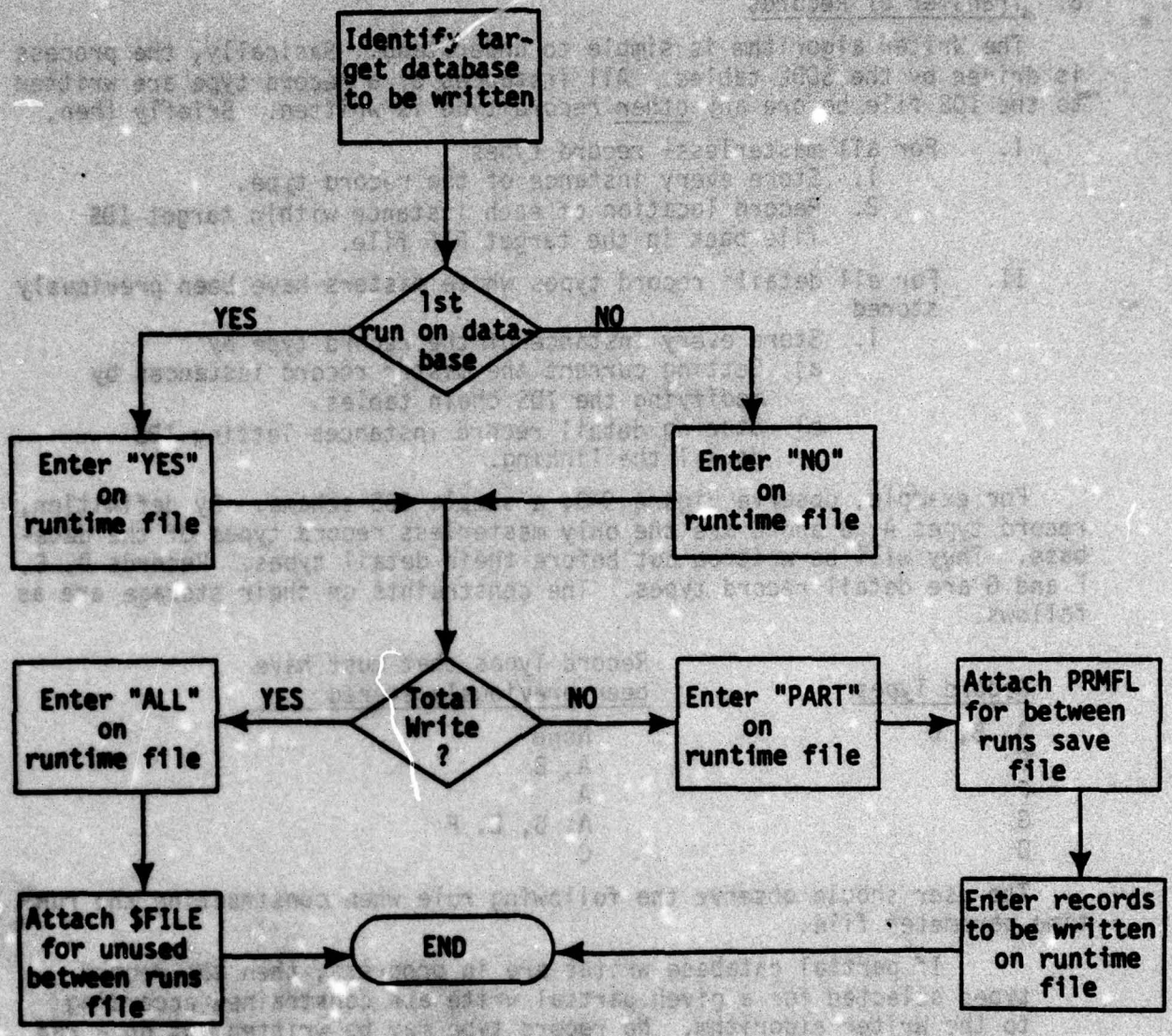


Figure 9-2
Run-time parameter file

B. Transfer of Records

The Writer algorithm is simple to understand. Basically, the process is driven by the SDDL tables. All instances of a record type are written to the IDS file before any other record type is written. Briefly then,

I. For all masterless[1] record types
   1. Store every instance of the record type.
   2. Record location of each instance within target IDS file back in the target RIF file.

II. For all detail[2] record types whose masters have been previously stored
   1. Store every instance of the record type by
      a) Setting current the master record instances by modifying the IDS chain tables.
      b) Storing detail record instances letting IDS do all the linking.

For example, observe Figure 9-3, a sample IDS schema. By definition, record types A, B and C are the only masterless record types of the database. They will be written out before their detail types. Records D, E, F and G are detail record types. The constraints on their storage are as follows.

Record Types	Record Types that must have been previously stored
A, B, C	None
E	A, B
F	A
G	A, B, E, F
D	C

The user should observe the following rule when constructing the run-time parameter file.

If partial database writes are in progress, then the record types selected for a given partial write are constrained according to the Writer algorithm. No record type may be written out on a run unless all of its master record types were previously written on prior runs or will be written on the current run. From a diagrammatic view, the target IDS database must be written in a top down fashion, starting with the masterless records. Figure 9-4 shows sample correct and incorrect partial database write sequences. Remember that the QUTI activity is used only for partial database write #1.

---

[1] A masterless record type is defined to be an IDS record that is either
   a) a CHAIN DETAIL of zero chains.
   b) a CHAIN DETAIL of only the CALC CHAIN.

[2] A detail record type is defined to be an IDS record that is a 98 CHAIN DETAIL of one or more non-CALC chains.

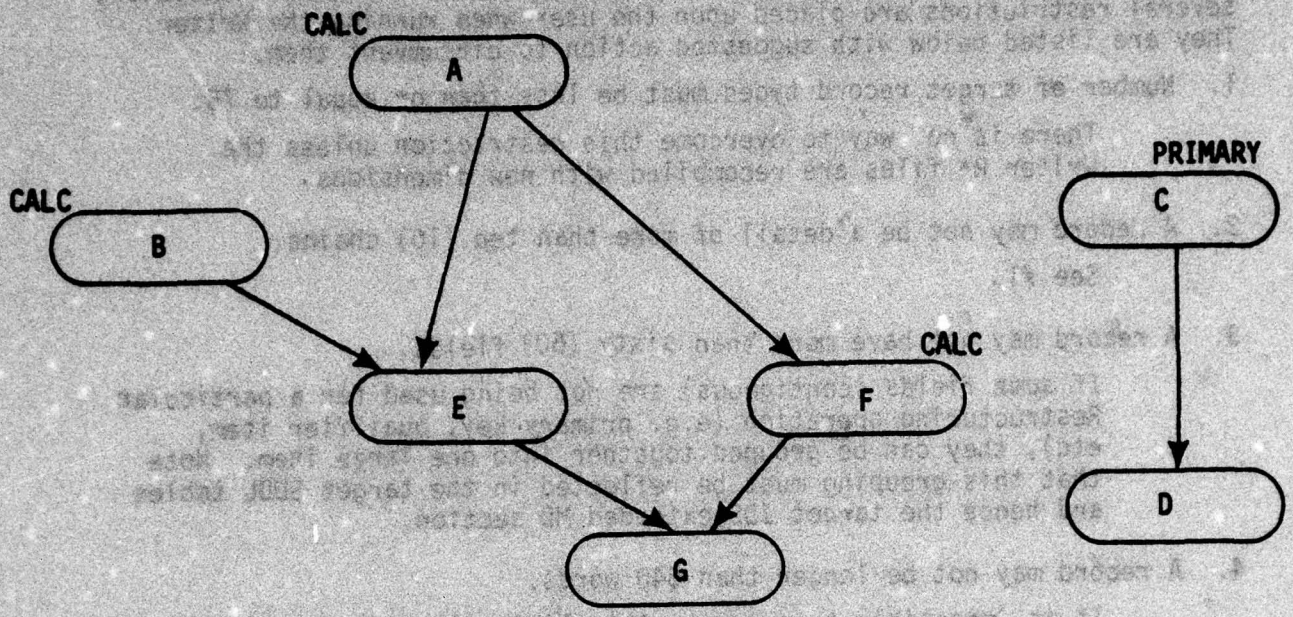


Figure 9-3
Sample IDS Schema

	Correct	Correct	Incorrect
Partial Run #1	A, B, E	B, C, D	A, B, D (D is a detail of C which has not been stored)
Partial Run #2	F, G	A, F	G (records E and F are not being written this run and were not written on run #1)
Partial Run #3	C	E, G	E, F, C
Partial Run #4	D		

Figure 9-4
Partial Database Write Sequences

9.2.5 Processing Limitations on Writer

Due to memory constraints in the development of the Data Translator, several restrictions are placed upon the user when running the Writer. They are listed below with suggested action to circumvent them.

1. Number of target record types must be less than or equal to 75.

   There is no way to overcome this restriction unless the Writer R* files are recompiled with new dimensions.

2. A record may not be a detail of more than ten (10) chains.

   See #1.

3. A record may not have more than sixty (60) fields.

   If some fields (contiguous) are not being used for a particular Restructuring operation (e.g. primary key, qualifier item, etc), they can be grouped together into one large item. Note that this grouping must be reflected in the target SDDL tables and hence the target IDS extended MD section.

4. A record may not be longer than 640 words.

   It is impossible to overcome this limitation; do not design target records larger than 640 words.

5. An item may not be longer than 255 characters.

   This is an ADBMS restriction. To avoid it, subdivide the "long" item into smaller (< 255) items. Subject to restriction #3 above.

6. Contained-in-repeating groups, match-key and phantom pointer relations cannot be produced in the target database.

   None of the above constructs should be in the target database. Such features are not under the control of IDS and are therefore undesirable.

7. The number of masterless record types may not exceed fifty (50).

   Short of redesigning the target database, there is no way to avoid this limitation except to recompile the Writer R* file.

9.3 Complete JCL to Execute the Writers

Figure 9-5 is the complete listing of control cards necessary to execute the Writer in one job. The sequence of control cards is available on the SYSGEN tape and must be modified according to which files the user is supplying for a given run. User changes to the control card sequences are listed below by line number.

Line	Description
0100	The PRMFL for the target RIF database produced by the Restructurer
0110	The size of the target RIF in random LINKS. Note that the target RIF size must be a multiple of 12 LLINKS or activity 5 will abnormally terminate within UTILITY.
0200	Target IDS database file. It must have filecode of Any subfiles must have filecodes in sequence from X1,X2...Xn.
0220	IDS INITIAL directives must be supplied for each subfile present. The entire range of pages must be initialized.
0270	Source code file from SYSGEN tape of part 1 of the COBOL-Writer Main program. Must be an ASCII file.
0280	ASCII file containing a legal target IDS MD section that was used as a base for the extended target MD section processed subsequently by the IDS Analyzer.
0290	Source code file from SYSGEN tape of part 2 of the COBOL-Writer Main program. Must be an ASCII file.
0340	The R* file of the Writer available on the SYSGEN tape.
0360	Time restrictions must be entered. Time restriction is computed using a guideline of 13,300 record instances/CPU hour.
0400	PRMFL containing the target SDDL tables produced by the IDS Analyzer.
0420-0430	Run-time parameter specifications containing the name of the database being written and any information regarding partial database writes. See Section 9.7 for exact format.
0440-0510	Either a $FILE 16,... or a $PRMFL 16... depending on whether a between runs save file is needed for this execution. All runs for a partial database write require a permanent between runs save file.

```
0010    $       IDENT
0020    $       NOTE    ************************************************************
0030    $       NOTE    *              EXECUTE THE WRITER JCL                      *
0040    $       NOTE    *                                                          *
0060    $       NOTE    *    UTILITY COPY OF TARGET RIF INTO TEMP. FILE            *
0070    $       NOTE    ************************************************************
0080    $       UTILITY
0090    $       FUTIL   A1,A2,RCOPY/1F/
0100    $       PRMFL   A1,R,R,TARGET RIF FILE
0110    $       FILE    A2,X1S,SIZE OF TARGET RIF IN LINKS
0120    $       NOTE    ************************************************************
0130    $       NOTE    *     IF THIS IS THE FIRST RUN ON A TARGET                 *
0140    $       NOTE    *     DATABASE, THEN INCLUDE THIS ACTIVITY,                *
0150    $       NOTE    *     ELSE SKIP DOWN TO THE COMPILATION                    *
0160    $       NOTE    *                                                          *
0170    $       NOTE    *       INITIALIZE TARGET IDS DATABASE                     *
0180    $       NOTE    ************************************************************
0190    $       PROGRAM QUTI
0200    $       PRMFL   X1,REC,R,TARGET IDS DATABASE
0210    $       DATA    I*
0220    IDS     INITIAL PAGE RANGE OF TARGET DATABASE
0230    $       NOTE    ************************************************************
0240    $       NOTE    *     COMPILE THE MD SECTION INTO THE WRITER               *
0250    $       NOTE    ************************************************************
0260    $       IDS     NDECK
0270    $       SELECTA COBOL-WRITER SOURCE CODE PT. 1
0280    $       SELECTA TARGET IDS MD SECTION
0290    $       SELECTA COBOL-WRITER SOURCE CODE PT. 2
0300    $       NOTE    ************************************************************
0310    $       NOTE    *              EXECUTE THE WRITER                          *
0320    $       NOTE    ************************************************************
0330    $       EXECUTE NREST
0340    $       PRMFL   R*,R,S,WRITER R*
0350    $       FILE    H*,H1R,20R
0360    $       LIMITS  XX,52K,-2K,10000    USE APPROP. TIME REQMNTS
0370    $       FILE    02,X1S              TARGET RIF
0380    $       REMOTE  06                  EXECUTION REPORT
0390    $       REMOTE  07                  ERROR REPORT
0400    $       PRMFL   08,W,R,TARGET SDDL TABLES
0410    $       DATA    15                  USER INPUT PARAMETERS
0420    DATABASE NAME
0430    SPECIFICATION OF RECORDS TO BE WRITTEN
0440    $       NOTE    ------------------------------------------------------------
0450    $       NOTE    -     INCLUDE EITHER-
0460    $       NOTE    -   $ FILE 16,X2R,2L   IF ALL RECORDS ARE WRITTEN
0470    $       NOTE    -                          THIS RUN
0480    $       NOTE    -          OR-
0490    $       NOTE    -   $ PRMFL 16,W,S,BETWEEN RUNS SAVE FILE IF ONLY
0500    $       NOTE    -                  PARTIAL DATABASE WRITE IS MADE
0510    $       NOTE    ------------------------------------------------------------
0520    $       PRMFL   X1,REC,R,TARGET IDS DATABASE
0530    $       PRMFL   TR,R,R,TRANSLATOR LIBRARY
0540    $       NOTE    ************************************************************
0550    $       NOTE    *     GO TO ENDJOB IF UNSUCCESSFUL WRITER RUN              *
0560    $       NOTE    *       ELSE COPY UPDATED RIF BACK TO PRMFL                *
0570    $       NOTE    ************************************************************
0580    $       IF      /35,ENDJOB
0590    $       UTILITY
0600    $       FUTIL   A1,A2,RCOPY/1F/
0610    $       FILE    A1,X1R
0620    $       PRMFL   A2,W,R,TARGET RIF FILE
0630    $       ENDJOB
```

Line	Description
0520	Target IDS database, same as in line 0200. X1 is the required filecode. If subfiles are used then the filecodes must be in sequence from X1, X2, X3 ... Xn.
0530	Object time random library of Data Translator routines available from SYSGEN tape. Must be attached to filecode TR.
0620	Target RIF file name, same as line 0100.

Note that lines 0190 through 0220 are not included for partial database writes that are the second or subsequent runs on the given IDS database file.

9.4 Activity 1 - Utility Copy of Target RIF

```
$       IDENT
$       NOTE  ************************************************************
$       NOTE  *                EXECUTE THE WRITER JCL                    *
$       NOTE  *                                                          *
$       NOTE  *     UTILITY COPY OF TARGET RIF INTO TEMP. FILE           *
$       NOTE  ************************************************************
$       UTILITY
$       FUTIL   A1,A2.RCOPY/IF/
$       PRMFL   A1,R,R.TARGET RIF FILE
$       FILE    A2.X1S.SIZE OF TARGET RIF IN LINKS
```

Filecode	LUD	Description
A1	--	PRMFL of target RIF produced by the Restructurer
A2	X1S	Temporary file of size equivalent to the target RIF. Must be a random file.

User Parameters

None

Example of Output

Output is normal UTILITY output and hence is not shown.

Possible errors

The user should check report code 53 to see if the message "RCOPYD 1 FILES" is printed. Any other result is an error in control cards. Correct if necessary and rerun.

9.5 Activity 2 - Initialize Target IDS Database

```
$       NOTE    ************************************************
$       NOTE    *     IF THIS IS THE FIRST RUN ON A TARGET     *
$       NOTE    *     DATABASE. THEN INCLUDE THIS ACTIVITY.    *
$       NOTE    *     ELSE SKIP DOWN TO THE COMPILATION        *
$       NOTE    *                                              *
$       NOTE    *        INITIALIZE TARGET IDS DATABASE        *
$       NOTE    ************************************************
$       PROGRAM QUTI
$       PRMFL   X1,REC,R,TARGET IDS DATABASE
$       DATA    I*
IDS     INITIAL PAGE RANGE OF TARGET DATABASE
```

Filecode	LUD	Description
X1	--	Target IDS database file. If subfiles are being used then several filecodes X1,Xw,...Xn are required for each PRMFL
I*	--	Directives to initialize each IDS subfile page range. Only one INITIAL directive is required if subfiles are not present.

User Parameters

None

Examples of Output

Standard IDS QUTI output. It is therefore not presented here.

Possible Errors

1. Incorrect directive syntax, e.g. column rules not obeyed, "INITIAL" spelled wrong, etc.

   Action: Correct directive and rerun Writer.

2. Incorrect page range specified; the INITIAL directive must specify the page numbers of the IDS file (or subfile) as it was created.

   Action: Correct directive(s) and rerun Writer.

9.6 Activity 3 - Compile COBOL-Writer Main Program

```
$       NOTE    ************************************************
$       NOTE    *     COMPILE THE MD SECTION INTO THE WRITER   *
$       NOTE    ************************************************
$       IDS     NDECK
$       SELECTA COBOL-WRITER SOURCE CODE PT. 1
$       SELECTA TARGET IDS MD SECTION
$       SELECTA COBOL-WRITER SOURCE CODE PT. 2
```

Filecode	LUD	Description
S*, *S	--	Resultant file from $SELECTA-ing the COBOL-Writer Main program part I, target IDS MD section and COBOL-Writer Main program part II in that sequence. Used as source IDS program input to the IDS Translator. A $ FILE *3 may be required for large databases.

User Parameters

None

Examples of Output

Standard compilation output from IDS then COBOL. It is therefore not presented here.

Possible Errors

1. IDS errors - any possible errors that IDS can detect could appear. However, since the target MD section was used as a base for the target extended MD section (and hence target SDDL tables) any errors appearing this late in the sequence of translation could cause rerunning of the IDS Analyzer on the target, TDL Analyzer and Restructurer. The user should have noted in Section 3 the requirement that the base for the *extended MD section must be a legal MD section with no errors.* If all translation steps have been followed, especially the rules in Section 3, then no errors should occur in activity three of the Writer.

   Action: If any of the following errors occur, and the same error is present (but undetected) in the target IDS extended (with level 61s) MD section, then the user must backtrack and recreate the target SDDL tables (after fixing the target MD and extended MD sections) and hence the TDL Analyzer and Restructurer must be rerun.

   CHAIN errors (Missing masters, details, loops)

   RECORD errors (invalid TYPE, incorrect PRIMARY or CALC specification)

   The inordinately high cost of recovery from these errors makes their avoidance essential. Carefully following the rules presented in Section 3 and the IDS Programmers Guide is necessary.

2. COBOL errors - will be of the following two types:
   a) Errors which use a Writer reserved word as a data item in the target MD section. This error cannot be detected by the IDS Analyzer.
   b) COBOL Data Division errors; invalid PICTURES, misspelled keywords, missing punctuation or incorrect SIZE clauses. Not all of these errors could have been previously detected unless the user had followed the advice of Section 3 and compiled the target IDS MD section into a dummy COBOL program to detect any COBOL ERRORS.

Action: If the user chose a Writer reserved word as a data item name, change the name to an unused one and rerun the Writer. If a COBOL error occurred in the Data Division and that error is present in the extended target IDS MD section, then the target SDDL tables must be recreated, and the TDL Analyzer and Restructurer rerun. Again note the high cost of this error and the emphasis on avoidance.

9.7 Activity 4 - Execute the Writer

```
$       NOTE    ************************************************
$       NOTE    *              EXECUTE THE WRITER              *
$       NOTE    ************************************************
$       EXECUTE NREST
$       PRMFL   R*,R,S,WRITER R*
$       FILE    H*,H1R,20R
$       LIMITS  XX,52K,-2K,10000    USE APPROP. TIME REQMNTS
$       FILE    02,X1S              TARGET RIF
$       REMOTE  06                  EXECUTION REPORT
$       REMOTE  07                  ERROR REPORT
$       PRMFL   08,W,R,TARGET SDDL TABLES
$       DATA    15                  USER INPUT PARAMETERS
DATABASE NAME
SPECIFICATION OF RECORDS TO BE WRITTEN
$       NOTE    ------------------------------------------------
$       NOTE    -      INCLUDE EITHER-
$       NOTE    -   $ FILE 16,X2R,2L   IF ALL RECORDS ARE WRITTEN
$       NOTE    -                          THIS RUN
$       NOTE    -          OR-
$       NOTE    -   $ PRMFL 16,W,S,BETWEEN RUNS SAVE FILE IF ONLY
$       NOTE    -                  PARTIAL DATABASE WRITE IS MADE
$       NOTE    ------------------------------------------------
$       PRMFL   X1,REC,R,TARGET IDS DATABASE
$       PRMFL   TR,R,R,TRANSLATOR LIBRARY
```

Filecode	LUD	Description
B* (not shown)	--	Output from COBOL-IDS compile from activity three.
R*	--	Writer R* file from SYSGEN tape.
H*	HIR	Temporary random file for Loader H* file. Must be supplied so entry can be made in PAT.
02	XIS	Temporary RIF database from activity one.

Filecode	LUD	Description
06	--	Writer Execution Report
07	--	Writer error messages
08	--	PRMFL for the target SDDL tables
15	--	Runtime parameter file is inserted following the $DATA 15 card. See below for format.
16	(X2R)	Between runs save file. If a temporary file is being used (e.g., a total database write), the X2R is the LUD; otherwise a PRMFL is included here.
X1	--	Target IDS database file. If subfiles are used they must have filecodes X1, X2, X3... assigned in sequence.
TR	--	Translator library of object routines.

User Parameters

The user must supply run-time parameters guiding the Writer according to the following rules.

```
CARD 1      database-name
CARD 2      first-run        total/partial
CARDS 3-N   record-name
```

Rules:

1. database-name must be a legal database name of one of the five possible databases present in the target RIF. It must correspond to the database-name supplied during IDS Analyzer execution yielding the target SDDL tables. See Section 5.6 in which the IDS Analyzer run-time parameter file was created for the target database description(s).

2. first-run is either

   YES in columns 1-3 indicates that this is the first Writer execution to output this database.

   or

   NO in columns 1-2 indicates that this is a second or subsequent run in writing this database. "NO" is only applicable for partial database writes. If NO is specified, a permanent between runs save file must be supplied on filecode 16.

3. total/partial is either

   ALL in columns 7-9 indicates that all records for database name are to be written on this run. "ALL" cannot be specified if "NO" was used for first-run.

or

PART in columns 7-10 indicates that a partial database write is being made; the records to be written will be identified on cards 3 through n.

4. record-name is a legitimate IDS 01 record name from the target MD section. For each record type to be written, one record name per card must be supplied. The record name must start in column 1.

Example Use of Run-time Parameter File

Let Figure 9-6 be the schema for the STATES database. Because it is desired that this database be written out in two runs, partial database write is required. Figure 9-7 shows the run-time parameter files required for runs 1 and 2. The file "MICHIGAN/PPS.SF" is the between runs save file for the STATES database.

Example Output

The Writer output can be divided into two sections: the execution report and the error message report. An explanation of the execution report is provided below; error messages are summarized in Appendix E.

Writer execution is subdivided into three units within the execution report.

INITIALIZATION (Figure 9-8)

A summary of the user's run-time parameter file is placed at the top of the execution report. Results of any previous partial database writes (not shown in Figure 9-8) are also summarized. Records to be written on the current run are listed with the following attributes.

IDS RECORD NAME - The 01 record name from the target SDDL tables of the record to be written.

IDS RECORD TYPE - IDS's record type value from 1-999 that identifies the record type.

RIF NAME - The ADBMS name that corresponds to the full IDS record name of the target RIF record.

ADDR. OF RD - Run time address of the Record Definition record of the IDS Definition Structure produced by compiling the target IDS MD section.

MASTERLESS Store (Figure 9-9)

As noted in Section 9.2, the Writer algorithm stores Masterless record types first. In Figure 9-9, the CONGRESS and STATES-IN-UNION records are the only masterless record types for the NEW-PRESIDENTIAL database. TOTAL # INSTANCES STORED is, to the best of the Writer's knowledge, the count of the number of record instances successfully stored into the target. Its value should be compared to the output of the Restructurer to determine whether instances were lost in the transfer between the target RIF database and the target IDS file. If there is a difference, it should be accounted for in the Writer error report.

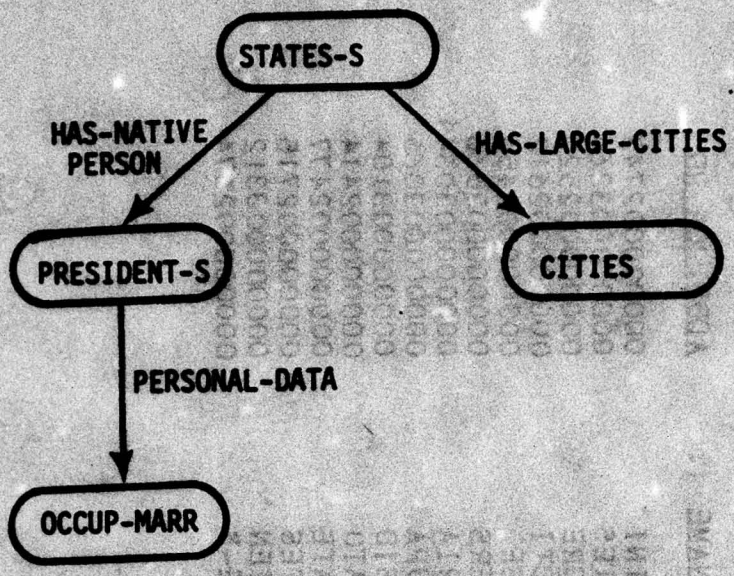


Figure 9-6
Sample Target Schema - STATES database

```
$       DATA    15                              USER INPUT PARAMETERS
STATES
YES     PART                            Run #1
STATES-S
CITIES
$       NOTE    ------------------------------------------------
$       NOTE    -       INCLUDE EITHER-
$       NOTE    -    $ FILE 16,X2R,2L    IF ALL RECORDS ARE WRITTEN
$       NOTE    -                            THIS RUN
$       NOTE    -            OR-
$       NOTE    -    $ PRMFL 16,W,S,BETWEEN RUNS SAVE FILE IF ONLY
$       NOTE    -                  PARTIAL DATABASE WRITE IS MADE
$       NOTE    ------------------------------------------------
$       PRMFL   16,W,S,MICHIGAN/PPS.SF
```

```
$       DATA    15                              USER INPUT PARAMETERS
STATES
NO      PART                            Run #2
PRESIDENT-S
OCCUP-MARR
$       NOTE    ------------------------------------------------
$       NOTE    -       INCLUDE EITHER-
$       NOTE    -    $ FILE 16,X2R,2L    IF ALL RECORDS ARE WRITTEN
$       NOTE    -                            THIS RUN
$       NOTE    -            OR-
$       NOTE    -    $ PRMFL 16,W,S,BETWEEN RUNS SAVE FILE IF ONLY
$       NOTE    -                  PARTIAL DATABASE WRITE IS MADE
$       NOTE    ------------------------------------------------
$       PRMFL   16,W,S,MICHIGAN/PPS.SF
```

Figure 9-7
Run-time parameter Files for
Partial Data Write of the STATES Database

```
**WRITER INVOKED**

THIS IS THE FIRST RUN OF THE WRITER ON DATABASE NEW-PRESIDENTIAL          DATE: NOV 12, 1976
   THE ENTIRE DATABASE WILL BE WRITTEN

 RECORDS TO BE WRITTEN ARE:

      IDS RECORD NAME              IDS RECORD TYPE     RIE NAME     ADDR. OF RD

      ADMINISTRATION                       6           ADMINI      000000002743
      CITIES                               2           CITIES      000000003266
      CONGRESS                            11           CONGRE      000000002520
      ELECTION                             9           ELECTI      000000002612
      HOUSE                               13           HOUSE       000000002456
      LOSERS                              10           LOSERS      000000002566
      MARRIAGE-DATA                        5           MARRIA      000000003034
      OCCUPATION                           4           OCCUPA      000000003066
      PRESIDENT                            3           PRESID      000000003104
      RELATOR-PRES-CONG                   14           RELATO      000000002414
      SENATE                              12           SENATE      000000002477
      STATES-ADMITTED                      7           STATES      000000002715
      STATES-IN-UNION                      1           STATEN      000000003312
      VICE-PRESIDENT                       8           VICE-P      000000002674

*INITIALIZATION COMPLETE
```

Figure 9-8
Writer Initialization Output

```
PHASE 1 OF WRITER...STORE MASTERLESS RECORD TYPES

     ****BEGIN  STORING RECORD TYPE-CONGRESS

     ****FINISH STORING RECORD TYPE-CONGRESS                    TOTAL # INSTANCES STORED =     14

     ****BEGIN  STORING RECORD TYPE-STATES-IN-UNION

     ****FINISH STORING RECORD TYPE-STATES-IN-UNION             TOTAL # INSTANCES STORED =     19
```

Figure 9-9
Writer Phase 1 Output

```
     ****BEGIN  STORING RECORD TYPE-PRESIDENT                   3

 THE RECORD TYPE-PRESIDENT                    HAS ATTRIBUTES-
DETAIL OF CHAIN          SET NAME  OWNER RECORD  NXT  PRIOR   ADDR. OF MD
HAS-NATIVE-PERSON        HAS-NA      STATEN      63    0      000000003336

     ****FINISH STORING RECORD TYPE-PRESIDENT                   TOTAL # INSTANCES STORED =      4


     ****BEGIN  STORING RECORD TYPE-RELATOR-PRES-CONG           14

 THE RECORD TYPE-RELATOR-PRES-CONG            HAS ATTRIBUTES-
DETAIL OF CHAIN          SET NAME  OWNER RECORD  NXT  PRIOR   ADDR. OF MD
SERVED-WITH-CONGRESS     SERVES      PRESID      156    0      000000003130
SERVED-WITH-PRESIDENT    SERVED      CONGRE       20    0      000000002526

     ****FINISH STORING RECORD TYPE-RELATOR-PRES-CONG           TOTAL # INSTANCES STORED =     14
```

Figure 9-10
Partial Phase 2 Writer Output

DETAIL Store (Figure 9-10)

Once all the masterless records have been stored, the Writer stores the detail records. Each detail record has attributes that are printed for informational purposes. These include:

DETAIL OF CHAIN - The IDS chain name(s) of which the record is a detail.

SET NAME - The ADBMS set name within the target RIF which corresponds to the chain name.

OWNER RECORD - The ADBMS record name which is the owner record of the SET NAME. The detail record being stored is a member of SET NAME.

NXT - Displacement within the IDS record (in characters) of the Chain next pointer for the chain name on the same line.

PRIOR - Same as NXT except for the Chain prior pointer. Zero if chain is not LINKED PRIOR.

ADDR. OF MD. - Run-time address within the .IDS.. common block of the Master Definition record for the chain on the same line of output.

As before, the user should verify that no record instances are unaccounted for between the target RIF file and the target IDS database.

Possible errors

Figure 9-11 presents an example of two Writer errors. A complete list of all Writer errors, their causes and appropriate actions is available in Appendix E.

Note: If an error occurs in a Writer execution that is serious enough to warrant rerunning the Writer, and if the following two conditions both hold, then special action is necessary:

a) The execution of the Writer with the error was not the first run of a partial database write sequence.

b) At least one instance of the erroneous record type was successfully stored into the IDS database.

The special action is simply to start all over again with the first run of the partial database write sequence (and hence, including the QUTI activity);otherwise, multiple copies of the same record instance will end up in the target database.

```
ERROR.....THE RECORD- STATES  IDENTIFIED BY KEY- 000006001063  IS NOT AN INSTANCE OF SET- ADMITT

ERROR....GETREF HAD ADBMS ERROR OCCUR FOR DBKEY-000006001063 WHILE LOOKING FOR OWNER OF DETAIL TYPE-STATE
      OWNER = ADMINISET = ADMITT
TAIL INSTANCE IGNORED
      FIRST FIVE WORDS OF RECORD INVOLVED IN ABOVE ERROR-
000000000044316262466451312020000021112000000202020202020  000000MISSOURI  00A9 000
```

Figure 9-11
Sample Writer Error Output

Filecode	LUD	Description
A1	X1R	Temporary copy of RIF created in activity one and updated in activity four.
A2	--	Permanent RIF file for target database(s). Same file as in activity one.

User Parameters

None

Example of Output

Standard UTILITY output, hence not shown.

Possible Errors

Activity four of the Writer will turn on bit 35 of the PSW if no fatal Writer errors occurred. Otherwise, the $IF card will cause activity five to be skipped as there is no longer a need to UTILITY copy.

Any errors in activity five would be due to control card errors which unfortunately would necessitate rerunning the Writer if (and only when) the Writer run represents a partial database write other than the last.

10.0 VALIDATING OUTPUT

The purpose of validating the target database is to assure the user that the transformation specified by the TDL produced the desired results. Deviations are possible because of the presence of semantic data in the source database or possible omissions in specifying desired target groups and relations. Improper access path designation in the TDL may produce target relations as specified but cause the omission of some desired target instances.

To ensure that a transformation has been successful it is necessary to exhaustively check both relationships and instances present in the target database. Three methods are suggested for application in varying situations.

10.1 COBOL Application Programs

For databases in a medium-to-very-large range (from 20 pages up), application programs undoubtedly provide the best exhaustive check of the database and can be formatted for rapid visual verification. Programming in COBOL is the preferred method of validation if the resulting code has some recurring value in use to justify the implementation time.

In the case of a single use, however, WWDMS provides a shortcut that is equally effective and less time consuming.

10.2 IDS Database Dumps

For small databases (less than 20 pages) a reasonable alternative to programming for verification is to use the IDS dump utility and check the database manually. Minimal knowledge of IDS storage structures and substantial patience are the only requirements.

Some valuable side effects of this method are that the database designer sees the physical result of page ranging and input ordering; the space and overhead involved with set relationships; and the physical size of records.

10.3 WWDMS Queries

The WWDMS environment allows a user to retrieve and format information from an IDS database in a shorthand manner. Without going into the details of the query system its use allows access to any information in the database via almost any route available to a COBOL application program.[1]

The user writes an Application Definition File (ADF) which specifies access paths through the database. After it has been processed with a Dictionary (the COBOL MD section) the paths may be nested to produce exhaustive dumps of the chained relations. Output formatting is as flexible as COBOL programs in a simple command (LINE).

---

[1]Present WWDMS-T2 does not allow the "heading" of nested chains.

For purposes of exhaustive checks it is necessary to write the ADF such that all paths emanating from a given SYSTEM entry point be nested to the point where another SYSTEM entry point is encountered. In this fashion, all instances and relationships in the database may be formatted to to permit simple cross-checking of the output lists.

11.0 EXAMPLE TRANSLATION

This section shows by example, the steps which a user must perform for each function of the Data Translator. The example used is the restructuring of the "NEW-PRESIDENTIAL" database into a "STATES" database. Bachman diagrams for each are shown in Figure 11-1.

The user's first step is usually augmenting her/his IDS MD section for the source database with 61 level entries. This process is described in Section 3.0. The augmented MD section for the NEW-PRESIDENTIAL database is shown in Figure 11-2. Next, the user should create an empty source SDDL database file and run the IDS Analyzer with the augmented IDS MD section as input. Setting up the JCL for the IDS Analyzer is described in Section 5.0. The JCL for this example is shown in Figure 11-3.

The user should then perform a similar set of steps for the target database. The 61 level augmented IDS MD section is shown in Figure 11-4 while Figure 11-5 is the IDS Analyzer JCL.

The next two steps, running the Reader and producing the TDL tables database, are independent processes and can be performed in any order. Example JCL for running the Reader is shown in Figure 11-6. Setting up the Reader JCL is discussed in Section 7.0. The TDL description for the translation is shown in Figure 11-7. Writing TDL descriptions is discussed in Section 4.0. The JCL to run the TDL Analyzer is shown in Figure 11-8. A description of running the TDL Analyzer can be found in Section 6.0.

The next step is to run the Restructurer. The Restructurer JCL is shown in Figure 11-9; the JCL is discussed in Section 8.0.

The final step is to run the Writer. The Writer JCL is shown in Figure 11-10 and discussed in Section 9.0.

Files Used in Example

File name	Usage
MICHIGAN/NPRES.61	Source extended MD section
/IDSA1.RS	Phase 1 IDS Analyzer R*
/TRANS.LR	Translator Library
/NPRES.ST	Source SDDL tables
/SDDL.AT	SDDL tables DBTF
/INTRN.AT	Internal Work DBTF
/NPRES.PM	Run-time parameter file (IDS Analyzer source)
/IDSA2.RS	Phase 2 IDS Analyzer R*
/PPS.61	Target extended MD section
/PPS.ST	Target SDDL tables
/PPS.PM	Run-time parameter file (IDS Analyzer target)
/ACIDS.S1	Reader source IDS Accessor (part 1)
/ACIDS.S2	Reader source IDS Accessor (part 2)
/NPRES.MD	Source IDS MD section
/READR.RS	Reader (IDS) R*
/DRT.DB	Reader DRT database
/DRT.AT	Reader DRT DBTF
/INTERNAL	Reader internal work file

/NPRES.SR	Source RIF
/NPRES.ID	Source IDS database
/TDL.RS	TDL Analyzer R*
/TDL.PT	TDL Analyzer parsing tables
/NPPS.TD	New Presidential-States TDL description
/NPPS.TT	TDL tables
/SYSAC.RS	SYSACC R* (TDL Analyzer)
/COMP.RS	Compatibility R* (TDL Analyzer)
/TDUMP.RS	TDL dump routine R* (TDL Analyzer)
/PPS.TR	States target RIF
/WRKDB.AF	Restructurer work database
/RESTR.RS	Restructurer R*
/PPS.ID	Target IDS file
/COB1.S	Writer source code Main program part 1
/COB2.S	Writer source code Main program part 2
/WRITE.RS	Writer R*

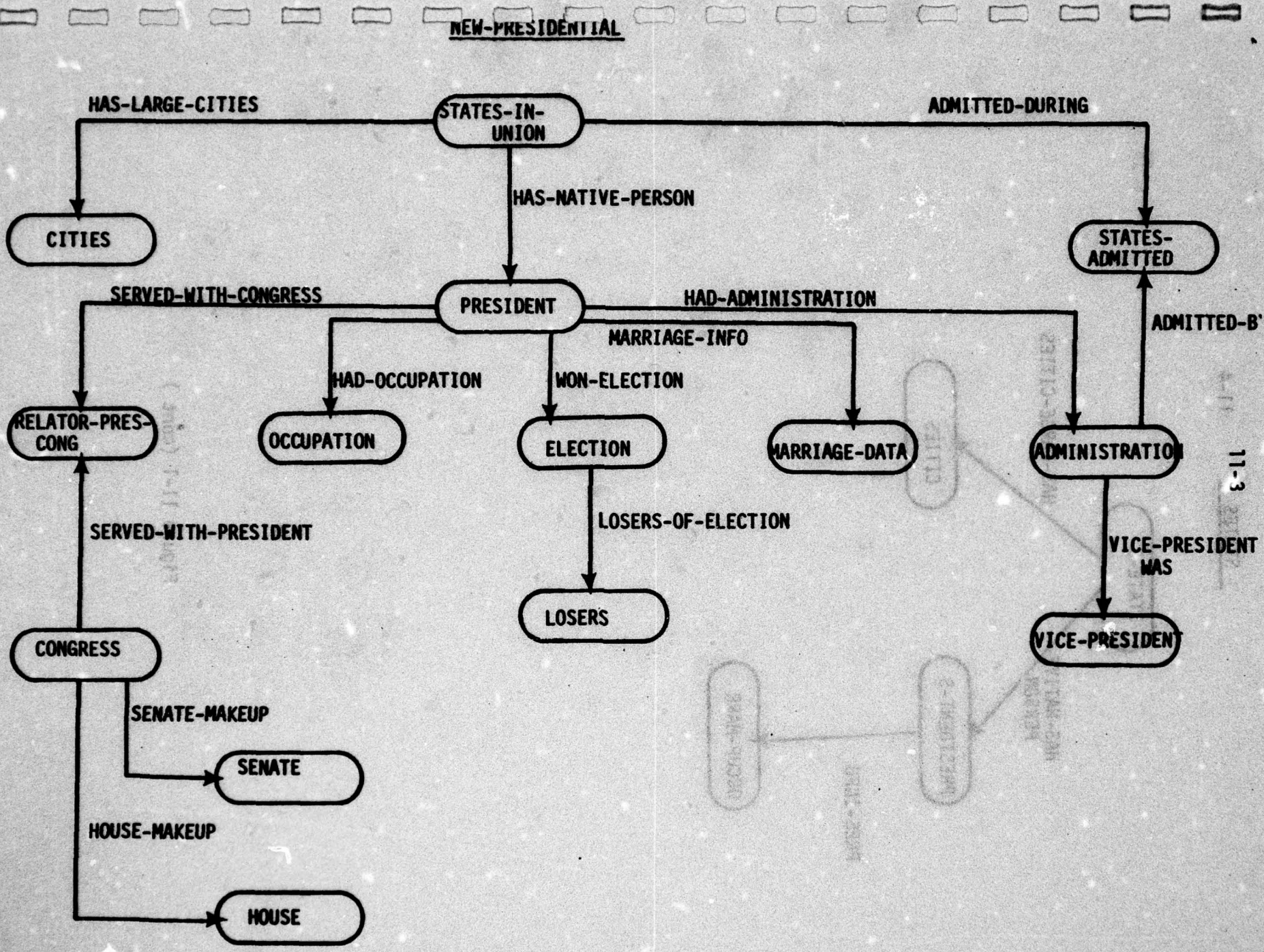

NEW-PRESIDENTIAL
HAS-LARGE-CITIES
STATES-IN-UNION
ADMITTED-DURING
CITIES
HAS-NATIVE-PERSON
STATES-ADMITTED
SERVED-WITH-CONGRESS
PRESIDENT
HAD-ADMINISTRATION
ADMITTED-B
MARRIAGE-INFO
HAD-OCCUPATION
WON-ELECTION
RELATOR-PRES-CONG
OCCUPATION
ELECTION
MARRIAGE-DATA
ADMINISTRATION
SERVED-WITH-PRESIDENT
LOSERS-OF-ELECTION
VICE-PRESIDENT WAS
LOSERS
VICE-PRESIDENT
CONGRESS
SENATE-MAKEUP
SENATE
HOUSE-MAKEUP
HOUSE

Figure 11-1

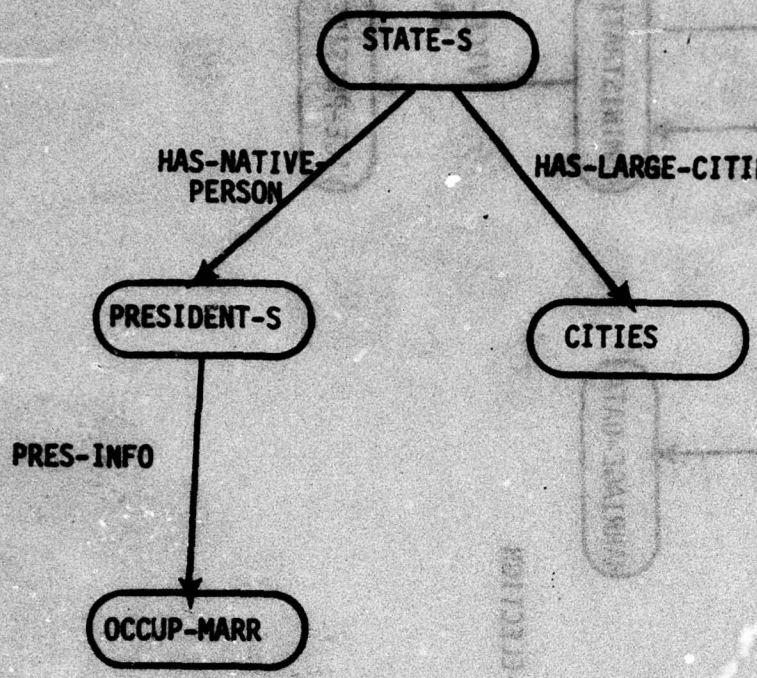


Figure 11-1 (cont.)

```
 IDS QUERY
*****************************************************************
*    EXTENDED SOURCE IDS MD SECTION FOR NEW-PRESIDENTIAL         *
*****************************************************************
*
 01   STATES-IN-UNION          TYPE IS 1 RETRIEVAL VIA CALC CHAIN
                               PAGE-RANGE IS 1 TO 6.
      02 STATE-NAME            PIC X(10).
      02 YEAR-ADMITTED         PIC X(4).
      02 CAPITAL               PIC X(10).
      02 AREA-SQ-MI            PIC 9(8) USAGE IS COMP-1.
      02 AREA-RANK             PIC 9(2) USAGE IS COMP-1.
      02 POPULATION            PIC 9(8) USAGE IS COMP-1.
      02 POP-RANK              PIC 9(2) USAGE IS COMP-1.
      02 ELECTORAL-VOTES       PIC 9(2) USAGE IS COMP-1.
      02 TRANSLATION-INFORMATION.
      61 P-K.
      61 G: STATES-IN-UNION,
      61 I: STATE-NAME.
 98   CALC CHAIN DETAIL
         RANDOMIZE ON STATE-NAME.
 98   HAS-NATIVE-PERSON CHAIN MASTER
         CHAIN-ORDER IS AFTER.
 98   HAS-LARGE-CITIES CHAIN MASTER
         CHAIN-ORDER IS SORTED.
 98   ADMITTED-DURING CHAIN MASTER
         CHAIN-ORDER IS AFTER.
*###############################################
*
 01   CITIES                   TYPE IS 2
                               RETRIEVAL VIA HAS-LARGE-CITIES CHAIN.
      02 CITY-NAME             PIC X(10).
      02 POPULATION-OF-CITY    PIC 9(10) USAGE IS COMP-1.
      02 TRANSLATION-INFORMATION SIZE 0.
      61 P-K.
      61 G: CITIES,
      61 I: CITY-NAME,
      61 POPULATION-OF-CITY.
      61 E-K-F: HAS-LARGE-CITIES.
 98   HAS-LARGE-CITIES CHAIN DETAIL
         DUPLICATES NOT ALLOWED
         ASCENDING KEY IS POPULATION-OF-CITY
         SELECT CURRENT MASTER.
*###############################################
*
 01 PRESIDENT                  TYPE IS 3
                               RETRIEVAL VIA CALC CHAIN
                               PAGE-RANGE IS 7 TO 13.
      02 LAST-NAME             PIC X(10).
      02 FIRST-NAME            PIC X(10).
      02 INIT                  PIC X(1).
      02 MONTH-BORN            PIC X(10).
      02 DAY-BORN              PIC X(2).
      02 YEAR-BORN             PIC X(4).
      02 HEIGHT                PIC X(10).
      02 PARTY-AFFILIATED      PIC X(10).
      02 COLLEGE               PIC X(10).
```

Figure 11-2

```
     02 ANCESTRY                PIC X(10).
     02 RELIGION                PIC X(10).
     02 MONTH-DIED              PIC X(10).
     02 DAY-DIED                PIC X(2).
     02 YEAR-DIED               PIC X(10).
     02 CAUSE-OF-DEATH          PIC X(10).
     02 FATHERS-NAME            PIC X(10).
     02 MOTHERS-NAME            PIC X(10).
     02 TRANSLATION-INFORMATION.
     61 P-K.
     61 G: PRESIDENT,
     61 I: LAST-NAME,
     61   FIRST-NAME,
     61   INIT.
 98  HAD-ADMINISTRATION CHAIN MASTER
       CHAIN-ORDER IS AFTER.
 98  WON-ELECTION CHAIN MASTER
       CHAIN-ORDER IS AFTER.
 98  SERVED-WITH-CONGRESS CHAIN MASTER
       CHAIN-ORDER IS AFTER.
 98  HAD-OCCUPATION CHAIN MASTER
       CHAIN-ORDER IS AFTER.
 98  MARRIAGE-INFO  CHAIN MASTER
       CHAIN-ORDER IS AFTER.
 98  CALC CHAIN   DETAIL
       RANDOMIZE ON LAST-NAME,  FIRST-NAME.
 98  HAS-NATIVE-PERSON  CHAIN DETAIL
       DUPLICATES NOT ALLOWED
       MATCH-KEY IS STATE-NAME
       SELECT UNIQUE MASTER
       LINKED TO MASTER.
*#########################################
 01  OCCUPATION   TYPE IS 4
                  RETRIEVAL VIA HAD-OCCUPATION CHAIN.
     02 TITLE-OF-JOB          PIC X(10).
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G: OCCUPATION,
     61 I:TITLE-OF-JOB.
     61 E-K-F:HAD-OCCUPATION.
 98  HAD-OCCUPATION  CHAIN DETAIL
       DUPLICATES NOT ALLOWED
       SELECT CURRENT MASTER.
*#########################################
*
 01  MARRIAGE-DATA            TYPE IS 5
                  RETRIEVAL VIA MARRIAGE-INFO CHAIN.
     02 WIVES-NAME            PIC X(10).
     02 MONTH-MARRIED         PIC X(10).
     02 DAY-MARRIED           PIC X(10).
     02 YEAR-MARRIED          PIC X(10).
     02 NUMBER-OF-CHILDREN     PIC X(10).
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G:MARRIAGE-DATA,
     61 I: WIVES-NAME.
     61 E-K-F:MARRIAGE-INFO.
 98  MARRIAGE-INFO  CHAIN DETAIL
```

Figure 11-2 (cont.)

```
        DUPLICATES NOT ALLOWED
        SELECT CURRENT MASTER.
*#########################################
 01  ADMINISTRATION   TYPE IS 6
                 RETRIEVAL VIA CALC CHAIN
                 PAGE-RANGE IS 14 TO 18.
     02 ADMINISTRATION-NUMBER PIC X(3).
     02 MONTH-INAUG           PIC X(10).
     02 DAY-INAUG             PIC X(2).
     02 YEAR-INAUG            PIC X(4).
     02 TRANSLATION-INFORMATION.
     61 P-K.
     61 G:ADMINISTRATION,
     61 I:
     61 ADMINISTRATION-NUMBER.
 98  CALC CHAIN DETAIL
        RANDOMIZE ON ADMINISTRATION-NUMBER.
 98  HAD-ADMINISTRATION CHAIN DETAIL
        SELECT UNIQUE MASTER
        DUPLICATES NOT ALLOWED
        MATCH-KEY IS LAST-NAME, MATCH-KEY IS FIRST-NAME
        LINKED TO MASTER.
 98  ADMITTED-STATES  CHAIN MASTER
        CHAIN-ORDER IS AFTER.
 98  VICE-PRESIDENT-WAS  CHAIN MASTER
        CHAIN-ORDER IS AFTER.
*#########################################
*
 01  STATES-ADMITTED   TYPE IS 7
                 RETRIEVAL VIA ADMITTED-STATES CHAIN.
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G:STATES-ADMITTED,
     61 E-K-F:
     61 ADMITTED-STATES,
     61 ADMITTED-DURING.
 98  ADMITTED-STATES CHAIN DETAIL
        LINKED TO MASTER
        DUPLICATES NOT ALLOWED
        SELECT CURRENT MASTER.
 98  ADMITTED-DURING  CHAIN DETAIL
        DUPLICATES NOT ALLOWED
        SELECT UNIQUE MASTER
        LINKED TO MASTER
        MATCH-KEY IS STATE-NAME.
*#########################################
*
 01  VICE-PRESIDENT   TYPE IS 8
                 RETRIEVAL VIA VICE-PRESIDENT-WAS CHAIN.
     02 VP-FIRST-NAME         PIC X(10).
     02 VP-LAST-NAME          PIC X(10).
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G:VICE-PRESIDENT,
     61 I:VP-FIRST-NAME,
     61    VP-LAST-NAME.
     61 E-K-F:
     61    VICE-PRESIDENT-WAS.
```

Figure 11-2 (cont.)

```
98  VICE-PRESIDENT-WAS CHAIN DETAIL.
       DUPLICATES NOT ALLOWED
       SELECT CURRENT MASTER.
*##########################################
*
 01  ELECTION     TYPE IS 9
                  RETRIEVAL VIA CALC CHAIN
                  PAGE-RANGE IS 19 TO 22.
     02 YEAR-OF-ELECTION          PIC X(5).
     02 WINNER                    PIC X(10).
     02 WINNING-PARTY             PIC X(10).
     02 WINNING-ELECTORAL-VOTES PIC X(3).
     02 TRANSLATION-INFORMATION.
     61 P-K.
     61 G:ELECTION,
     61 I:YEAR-OF-ELECTION.
 98  CALC CHAIN DETAIL
       RANDOMIZE ON YEAR-OF-ELECTION.
 98  WON-ELECTION CHAIN DETAIL
       SELECT UNIQUE MASTER
       MATCH-KEY IS LAST-NAME, MATCH-KEY IS FIRST-NAME
       LINKED TO MASTER.
 98  LOSERS-OF-ELECTIONS CHAIN MASTER
       CHAIN-ORDER IS AFTER.
*##########################################
*
 01  LOSERS     TYPE IS 10
                 RETRIEVAL VIA LOSERS-OF-ELECTIONS CHAIN.
     02 LOSER-NAME                PIC X(10).
     02 LOSER-PARTY               PIC X(10).
     02 LOSER-ELECTORAL-VOTES  PIC X(3).
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G:LOSERS,
     61 I:LOSER-NAME.
     61 E-K-F:
     61  LOSERS-OF-ELECTIONS.
 98  LOSERS-OF-ELECTIONS CHAIN DETAIL
       SELECT CURRENT MASTER
       DUPLICATES NOT ALLOWED.
*##########################################
*
 01  CONGRESS     TYPE IS 11
                  RETRIEVAL VIA CALC CHAIN
                  PAGE-RANGE IS 23 TO 33.
     02 CONGRESS-NUMBER           PIC X(3).
     02 TRANSLATION-INFORMATION.
     61 P-K.
     61 G: CONGRESS,
     61 I:CONGRESS-NUMBER.
 98  CALC CHAIN DETAIL
       RANDOMIZE ON CONGRESS-NUMBER.
 98  SENATE-MAKEUP  CHAIN MASTER
       CHAIN-ORDER IS AFTER.
 98  HOUSE-MAKEUP  CHAIN MASTER
       CHAIN-ORDER IS AFTER.
 98  SERVED-WITH-PRESIDENT  CHAIN MASTER
```

Figure 11-2 (cont.)

```
        CHAIN-ORDER IS AFTER.
*#########################################
*
 01  SENATE    TYPE IS 12
                RETRIEVAL VIA SENATE-MAKEUP CHAIN.
     02 SENATE-PARTY          PIC X(10).
     02 SENATORS              PIC X(2).
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G: SENATE,
     61 I: SENATE-PARTY,
     61    SENATORS.
     61 E-K-F:SENATE-MAKEUP.
 98  SENATE-MAKEUP CHAIN DETAIL
        SELECT CURRENT MASTER
        DUPLICATES NOT ALLOWED.
*#########################################
*
 01   HOUSE    TYPE IS 13
                RETRIEVAL VIA HOUSE-MAKEUP CHAIN.
     02 HOUSE-PARTY           PIC X(10).
     02 REPRESENTATIVES       PIC X(3).
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G:HOUSE,
     61 I:HOUSE-PARTY,
     61 REPRESENTATIVES.
     61 E-K-F: HOUSE-MAKEUP.
 98  HOUSE-MAKEUP CHAIN DETAIL
        SELECT CURRENT MASTER
        DUPLICATES NOT ALLOWED.
*#########################################
*
 01  RELATOR-PRES-CONG    TYPE IS 14
                RETRIEVAL VIA SERVED-WITH-CONGRESS CHAIN
                PAGE-RANGE IS 34 TO 35.
     02 TRANSLATION-INFORMATION SIZE 0.
     61 P-K.
     61 G:RELATOR-PRES-CONG,
     61 E-K-F:
     61 SERVED-WITH-CONGRESS,
     61 SERVED-WITH-PRESIDENT.
 98  SERVED-WITH-CONGRESS  CHAIN DETAIL
        LINKED TO MASTER
        SELECT UNIQUE MASTER
        MATCH-KEY IS LAST-NAME, MATCH-KEY IS FIRST-NAME
       DUPLICATES NOT ALLOWED.
 98  SERVED-WITH-PRESIDENT CHAIN DETAIL
        SELECT UNIQUE MASTER
        LINKED TO MASTER
         DUPLICATES NOT ALLOWED
        MATCH-KEY IS CONGRESS-NUMBER.
*#########################################
*#########################################
```

Figure 11-2 (cont.)

```
$       IDENT
$       NOTE    ************************************************************
$       NOTE    *      INITIALIZE QUERY DICTIONARY                         *
$       NOTE    ************************************************************
$       PROGRAM QUTI
$       FILE    A1,X1S,30R          QUERY DICTIONARY FILE
$       DATA    I*
IDS     INITIAL 1,360
$       DATA    .Q
IDS     CREATE  FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/
$       NOTE    ************************************************************
$       NOTE    *      CREATE IDS QUERY DICTIONARY                         *
$       NOTE    ************************************************************
$       IDS     NDECK
$       SELECTA MICHIGAN/NPRES.61
$       FILE    *3,X1S              QUERY DICTIONARY
$       DATA    .Q
IDS     CREATE  FC/*3/,BSSZ/360/,RNG/1,360/,INV/NO/
$       NOTE    ************************************************************
$       NOTE    *      EXECUTE IDS ANALYZER PHASE 1                        *
$       NOTE    ************************************************************
$       EXECUTE NREST
$       LIMITS  25,63K,-2K,30000
$       PRMFL   R*,R,R,MICHIGAN/IDSA1.RS
$       FILE    H*,X2R,50R
$       FILE    A1,X1R              QUERY DICTIONARY
$       DATA    .Q
IDS     CREATE  FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       PRMFL   03/Z1S,W,R,MICHIGAN/NPRES.ST
$       PRMFL   04,R,S,MICHIGAN/SDDL.AT
$       FILE    05,X3R              SCRATCH FILE
$       REMOTE  06                  INITIALIZATION OUTPUT
$       FILE    15,X4S,10R          INTERNAL WORK DATABASE
$       PRMFL   16,R,S,MICHIGAN/INTRN.AT
$       REMOTE  A3                  ERROR MESSAGES
$       DATA    A5
$       SELECTA MICHIGAN/NPRES.PM
$       NOTE    ************************************************************
$       NOTE    *      EXECUTE IDS ANALYZER PHASE 2                        *
$       NOTE    ************************************************************
$       EXECUTE
$       PRMFL   R*,R,S,MICHIGAN/IDSA2.RS
$       FILE    H*,H1R,10R
$       LIMITS  15,37K,-1K,15000
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       FILE    03,Z1R              SDDL TABLES
$       REMOTE  06                  SDDL DUMP AND ERROR MESSAGES
$       REMOTE  07                  SDDL DUMP
$       FILE    15,X4R              INTERNAL WORK DATABASE
$       ENDJOB
```

Figure 11-3

```
IDS   ( QUERY
*****************************************************************
*              EXTENDED TARGET IDS MD SECTION FOR STATES        *
*****************************************************************
*
 01  STATES-S          TYPE IS 1 RETRIEVAL VIA CALC CHAIN
                              PAGE-RANGE IS 1 TO 6'.
     02 STATE-NAME            PIC X(10).
     02 YEAR-ADMITTED         PIC X(4).
     02 CAPITAL               PIC X(10)'.
     02 AREA-SQ-MI            PIC 9(8) USAGE IS COMP-1'.
     02 AREA-RANK             PIC 9(2) USAGE IS COMP-1'.
     02 POPULATION            PIC 9(8) USAGE IS COMP-1'.
     02 POP-RANK              PIC 9(2) USAGE IS COMP-1'.
     02 ELECTORAL-VOTES       PIC 9(2) USAGE IS COMP-1'.
        02 ADMITTED-BY-ADMIN
        02 TRANSLATION-INFORMATION.
        01 P-K.
        61 G: STATES-S,
        61 I: STATE-NAME'.
 98  HAS-NATIVE-PERSON CHAIN MASTER
        CHAIN-ORDER IS AFTER'.
 98  CALC CHAIN DETAIL
        RANDOMIZE ON STATE-NAME'.
 98  HAS-LARGE-CITIES CHAIN MASTER
        CHAIN-ORDER IS SORTED.
*##############################################
*
 01  CITIES                TYPE IS 2
                              RETRIEVAL VIA REFCODE-CITIES FIELD.
        02 REFCODE-CITIES
     02 CITY-NAME             PIC X(10)'.
     02 POPULATION-OF-CITY    PIC 9(10) USAGE IS COMP-1.
        02 TRANSLATION-INFORMATION'.
        61 P-K'.
        61 G: CITIES,
        61 I: CITY-NAME,
        01 POPULATION-OF-CITY'.
        61 E-K-F:
        61     HAS-LARGE-CITIES.
 98  HAS-LARGE-CITIES CHAIN DETAIL
```

Figure 11-4

```
        DUPLICATES NOT ALLOWED
        ASCENDING KEY IS CITY-NAME
        SELECT CURRENT MASTER.
*#########################################
 01 PRESIDENT-S              TYPE IS 3
                            RETRIEVAL VIA REFCODE-P-S FIELD.
          02 REFCODE-P-S     PIC 9(8).
          02 PRES-S-LNAME    PIC X(10).
          02 PRES-S-FNAME    PIC X(10).
          02 PRES-S-MNAME    PIC X(1).
          02 TRANSLATION-INFORMATION.
          01 P-K.
          61 G: PRESIDENT-S,
          61 I: PRES-S-LNAME,
          61    PRES-S-FNAME,
          61    PRES-S-MNAME.
 98 HAS-NATIVE-PERSON CHAIN DETAIL
          DUPLICATES NOT ALLOWED
          SELECT CURRENT MASTER.
 98 PRES-INFO CHAIN MASTER
          CHAIN-ORDER IS AFTER.
*#########################################
 01  OCCUP-MARR    TYPE IS 4
                   RETRIEVAL VIA PRES-INFO CHAIN.
      02 TITLE-OF-JOB          PIC X(10).
      02 WIVES-NAME            PIC X(10).
      02 MONTH-MARRIED         PIC X(10).
      02 DAY-MARRIED           PIC X(10).
      02 YEAR-MARRIED          PIC X(10).
      02 NUMBER-OF-CHILDREN     PIC X(10).
          02 TRANSLATION-INFORMATION SIZE 0.
          61 P-K.
          6.1 G: OCCUP-MARR,
          6.1 I: TITLE-OF-JOB,
          6.1    WIVES-NAME.
          61     E-K-F:PRES-INFO.
 98 PRES-INFO CHAIN DETAIL
          SELECT CURRENT MASTER.
```

Figure 11-4 (cont.)

```
$       IDENT
$       NOTE    ************************************************************
$       NOTE    *      INITIALIZE QUERY DICTIONARY                         *
$       NOTE    ************************************************************
$       PROGRAM QUTI
$       FILE    A1,X1S,30R        QUERY DICTIONARY FILE
$       DATA    I*
IDS     INITIAL 1,360
$       DATA    .Q
IDS     CREATE  FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/
$       NOTE    ************************************************************
$       NOTE    *      CREATE IDS QUERY DICTIONARY                         *
$       NOTE    ************************************************************
$       IDS     NDECK
$       SELECTA MICHIGAN/PPS.61
$       FILE    *3,X1S            QUERY DICTIONARY
$       DATA    .Q
IDS     CREATE  FC/*3/,BSSZ/360/,RNG/1,360/,INV/NO/
$       NOTE    ************************************************************
$       NOTE    *      EXECUTE IDS ANALYZER PHASE 1                        *
$       NOTE    ************************************************************
$       EXECUTE NREST
$       LIMITS  25,63K,-2K,30000
$       PRMFL   R*,R,R,MICHIGAN/IDSA1.RS
$       FILE    H*,X2R,50R
$       FILE    A1,X1R            QUERY DICTIONARY
$       DATA    .Q
IDS     CREATE  FC/A1/,BSSZ/360/,RNG/1,360/,INV/NO/
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       PRMFL   03/Z1S,W,R,MICHIGAN/PPS.ST
$       PRMFL   04,R,S,MICHIGAN/SDDL.AT
$       FILE    05,X3R            SCRATCH FILE
$       REMOTE  06                INITIALIZATION OUTPUT
$       FILE    15,X4S,10R        INTERNAL WORK DATABASE
$       PRMFL   16,R,S,MICHIGAN/INTRN.AT
$       REMOTE  A3                ERROR MESSAGES
$       DATA    A5
$       SELECTA MICHIGAN/PPS.PM
$       NOTE    ************************************************************
$       NOTE    *      EXECUTE IDS ANALYZER PHASE 2                        *
$       NOTE    ************************************************************
$       EXECUTE
$       PRMFL   R*,R,S,MICHIGAN/IDSA2.RS
$       FILE    H*,H1R,10R
$       LIMITS  15,37K,-1K,15000
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       FILE    03,Z1R            SDDL TABLES
$       REMOTE  06                SDDL DUMP AND ERROR MESSAGES
$       REMOTE  07                SDDL DUMP
$       FILE    15,X4R            INTERNAL WORK DATABASE
$       ENDJOB
```

Figure 11-5

```
$       IDENT
$       NOTE,    <-------------------------------------------------->
$       NOTE.    <  ACCESSOR IDS-COBOL COMPILATION                  >
$       NOTE     <-------------------------------------------------->
$       IDS ,    NDECK
$       SELECTA  MICHIGAN/ACIDS.S1
$       SELECTA  MICHIGAN/NPRES.MD
$       SELECTA  MICHIGAN/ACIDS.S2
$       NOTE     <-------------------------------------------------->
$       NOTE     <  EXECUTE THE READER                              >
$       NOTE.    <-------------------------------------------------->
$       EXECUTE
$       PRMFL    R*,R,S,MICHIGAN/READR.RS      READER R* FILE
$       FILE,    H*,Y1R,15R
$       LIMITS   30,60K,-5K,50000              SET CORE AND CPU LIMITS
$       PRMFL    TR,R,R,MICHIGAN/TRANS.LR      TRANSLATION LIBRARY
$       PRMFL    01,W,R,MICHIGAN/DRT.DB        DRT DATABASE
$       PRMFL    02,W,R,MICHIGAN/DRT.AT        DRT ADBMS FILE
$       PRMFL    04,W,S,MICHIGAN/INTERNAL      READER INTERNAL FILE
$       DATA     05
    FIRST   LAST   NAME=NEW-PRESIDENTIAL   RECORDS=ALL
$       REMOTE   06                            REPORT OUTPUT
$       PRMFL    07,W,R,MICHIGAN/NPRES.ST      SDDL TABLE DATABASE
$       PRMFL    09,W,R,MICHIGAN/NPRES.SR      SOURCE RIF
$       FILE     10,R2R,10L                    SCRATCH FILE
$       FILE     12,R3R,10L                    SCRATCH FILE
$       FILE     13,NULL
$       PRMFL    X1,R,R,MICHIGAN/NPRES.ID      SOURCE IDS DATABASE
$       ENDJOB
```

Figure 11-6

```
/*   NEW PREZ TO PREZ PARTS STATES  */
/*  $L  */
TARGET RECORD CITIES
    TDLAP CITI
        SOURCE RECORD STATES-IN-UNION ACCESS VIA CALC-STATES-IN-UNION
            SOURCE RECORD CITIES ACCESS VIA HAS-LARGE-CITIES
                ACTUAL DATA IN ORDER
                SET SIGNIFICANT DATA BY NAME
TARGET RECORD OCCUP-MARR
    TDLAP OCCU
        SOURCE RECORD PRESIDENT ACCESS VIA CALC-PRESIDENT
            FIRST-NAME ASSIGN TO PRES-S-FNAME<PRES-INFO>
            INIT ASSIGN TO PRES-S-MNAME<PRES-INFO>
            LAST-NAME ASSIGN TO PRES-S-LNAME<PRES-INFO>
                SOURCE RECORD OCCUPATION ACCESS VIA HAD-OCCUPATION
                    TITLE-OF-JOB ASSIGN TO TITLE-OF-JOB
                SOURCE RECORD MARRIAGE-DATA ACCESS VIA
                        MARRIAGE-INFO FROM PRESIDENT
                    WIVES-NAME ASSIGN TO WIVES-NAME
                    MONTH-MARRIED ASSIGN TO MONTH-MARRIED
                    DAY-MARRIED ASSIGN TO DAY-MARRIED
                    YEAR-MARRIED ASSIGN TO YEAR-MARRIED
                    NUMBER-OF-CHILDREN ASSIGN TO NUMBER-OF-CHILDREN
TARGET RECORD PRESIDENT-S
    TDLAP PRES
        SOURCE RECORD PRESIDENT ACCESS VIA CALC-PRESIDENT
                SET SIGNIFICANT DATA BY NAME
            FIRST-NAME ASSIGN TO PRES-S-FNAME
            INIT ASSIGN TO PRES-S-MNAME
            LAST-NAME ASSIGN TO PRES-S-LNAME
TARGET RECORD STATES-S
    TDLAP STAT
        SOURCE RECORD STATES-IN-UNION ACCESS VIA CALC-STATES-IN-UNION
            STATE-NAME ASSIGN TO STATE-NAME
            YEAR-ADMITTED ASSIGN TO YEAR-ADMITTED
            CAPITAL ASSIGN TO CAPITAL
            AREA-SQ-MI ASSIGN TO AREA-SQ-MI
            AREA-RANK ASSIGN TO AREA-RANK
            POPULATION ASSIGN TO POPULATION
            POP-RANK ASSIGN TO POP-RANK
            ELECTORAL-VOTES ASSIGN TO ELECTORAL-VOTES
        SOURCE RECORD STATES-ADMITTED ACCESS VIA ADMITTED-DURING
            ACCEPT IF NULL
                SOURCE RECORD ADMINISTRATION ACCESS VIA
                        ADMITTED-STATES
                    ACCEPT IF NULL
                ADMINISTRATION-NUMBER ASSIGN TO
                        ADMITTED-BY-ADMIN
                    NULL VALUE = ' '
 EOF
```

Figure 11-7

```
$       IDENT
$       NOTE    ***********************************************
$       NOTE    *                                             *
$       NOTE    *    COPY SOURCE AND TARGET SDDL TABLES       *
$       NOTE    *                                             *
$       NOTE    ***********************************************
$       UTILITY
$       FUTIL   14,15,RWD/14/,RCOPY/1F/
$       PRMFL   14,R,R,MICHIGAN/NPRES.ST
$       FILE    15,X1S,4R
$       FUTIL   16,17,RWD/16/,RCOPY/1F/
$       PRMFL   16,R,R,MICHIGAN/PPS.ST
$       FILE    17,X2S,4R
$       NOTE    ***********************************************
$       NOTE    *                                             *
$       NOTE    *          EXECUTE TDL ANALYZER               *
$       NOTE    *                                             *
$       NOTE    ***********************************************
$       EXECUTE NREST
$       LIMITS  ,48K,-3K
$       PRMFL   R*,R,S,MICHIGAN/TDL.RS
$       FILE    H*,Z1R,40R
$       PRMFL   04,R,S,MICHIGAN/TDL.PT
$       DATA    02,,COPY
$       SELECTA MICHIGAN/NPPS.TD,
$       ENDCOPY
$       NOTE    ANALYZER OUTPUT IS ON FILE CODE 06
$       FILE    07,X1R
$       FILE    09,X2R
$       PRMFL   11,W,R,MICHIGAN/NPPS.TT
$       PRMFL   12,R,S,ADBMS TABLE FILE FOR TDL TABLES
$       FILE    13,NULL
$       FILE    14,NULL
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       IF      18,IF           IF TDL ERRORS THEN SKIP TO DUMP
$       NOTE    ***********************************************
$       NOTE    *                                             *
$       NOTE    *     POST PROCESS I - SYSACC BUILDER         *
$       NOTE    *                                             *
$       NOTE    ***********************************************
$       EXECUTE NREST
$       LIMITS  ,33K,-3K
$       PRMFL   R*,R,S,MICHIGAN/SYSAC.RS
$       PRMFL   07,W,R,MICHIGAN/NPPS.TT
$       FILE    08,X2S
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       NOTE    ***********************************************
$       NOTE    *                                             *
$       NOTE    *  POST PROCESS II - COMPATIBILITY BUILDER    *
$       NOTE    *                                             *
$       NOTE    ***********************************************
```

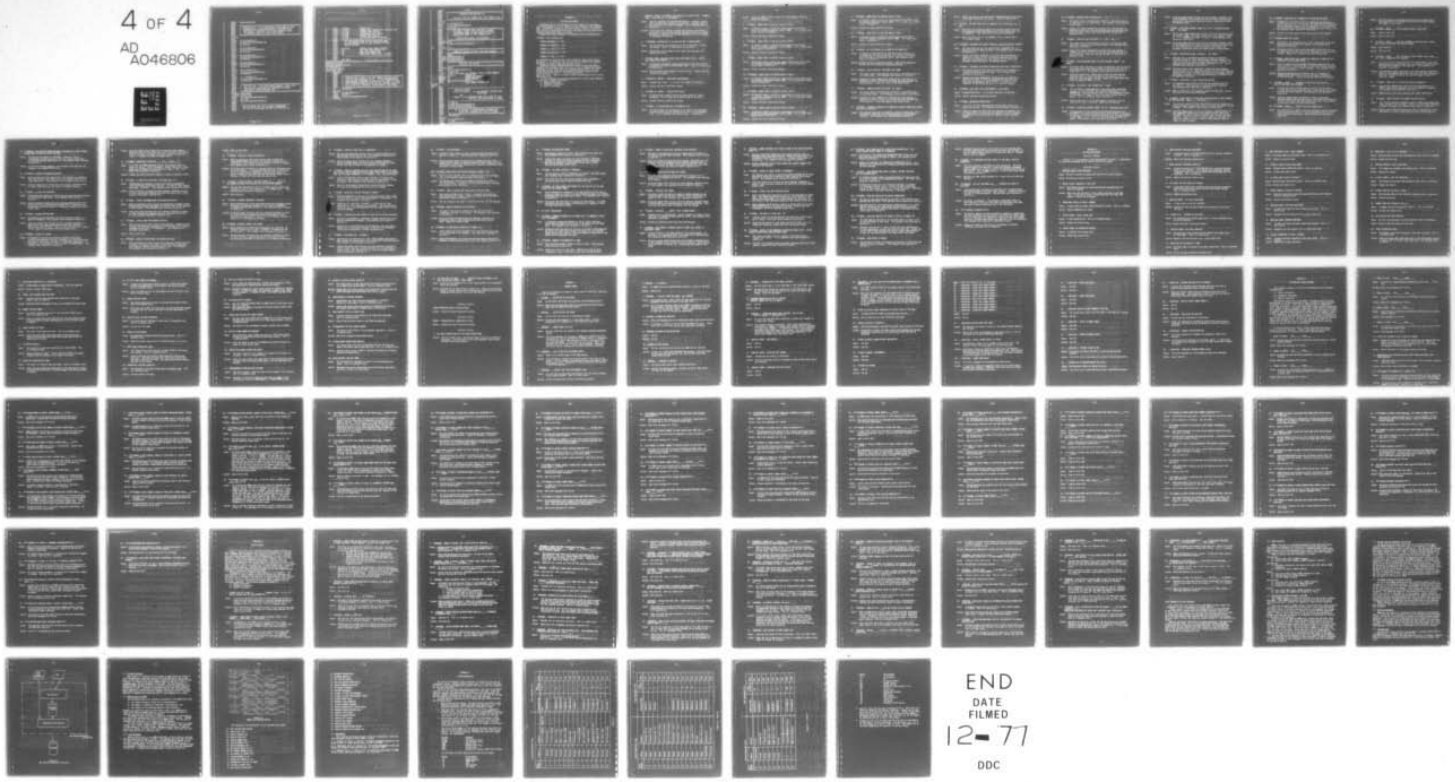
Figure 11-8

```
$       EXECUTE NREST
$       LIMITS  ,33K,-3K
$       PRMFL   R*,R,S,MICHIGAN/COMP.RS
$       PRMFL   07,W,R,MICHIGAN/NPPS.TT
$       FILE    08,X2S
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       NOTE    ****************************************
$       NOTE    *                                      *
$       NOTE    *            TDL TABLE DUMPER          *
$       NOTE    *                                      *
$       NOTE    ****************************************
$       IF      19,ENDJOB       SKIP DUMP IF NOT REQUESTED
$       EXECUTE NDUMP
$       LIMITS  ,25K,-3K
$       PRMFL   R*,R,S,MICHIGAN/TDUMP.RS
$       PRMFL   02,R,R,MICHIGAN/NPPS.TT
$       NOTE    DUMP OUTPUT IS ON FILE CODE 06
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       ENDJOB
```

Figure 11-8 (cont.)

AD-A046 806 MICHIGAN UNIV ANN ARBOR GRADUATE SCHOOL OF BUSINESS--ETC F/G 9/2
DATA TRANSLATOR VERSION IIA, RELEASE 2 USER MANUAL REVISION 1.(U)
MAR 77 E KINTZER, J BODWIN, W COOL DCA100-75-C-0064
UNCLASSIFIED WP-DT-3.4 CCTC-CSM-UM-236-77-REV-1 NL

4 OF 4
AD
A046806



END
DATE
FILMED
12-77
DDC

```
$       IDENT   MICHO,MICHIGAN
$       NOTE
$       NOTE    ************************************************************
$       NOTE    *  ACTIVITY 01 -- ALL DATABASES TO BE USED BY THE          *
$       NOTE    *  RESTRUCTURER ARE COPIED FIRST TO PROTECT THE            *
$       NOTE    *  ORIGINALS.  BE SURE TO PUT THE CORRECT SIZES            *
$       NOTE    *  (IN RANDOM LINKS) ON THE $FILE CARDS.                   *
$       NOTE    ************************************************************
$       UTILITY
$       NOTE
$       FUTIL   01,02,RCOPY/1F/
$       PRMFL   01,R,R,MICHIGAN/NPRES.SR
$       FILE    02,X0S,6R
$       NOTE
$       FUTIL   03,04,RCOPY/1F/
$       PRMFL   03,W,R,MICHIGAN/PPS.TR
$       FILE    04,X1S,3R
$       NOTE
$       FUTIL   05,06,RCOPY/1F/
$       PRMFL   05,R,R,MICHIGAN/NPPS.TT
$       FILE    06,X2S,3R
$       NOTE
$       FUTIL   07,08,RCOPY/1F/
$       PRMFL   07,R,R,MICHIGAN/NPPS.TT
$       FILE    08,X3S,3R
$       NOTE
$       FUTIL   09,10,RCOPY/1F/
$       PRMFL   09,R,R,MICHIGAN/PPS.ST
$       FILE    10,X4S,2R
$       NOTE
$       FUTIL   11,12,RCOPY/1F/
$       PRMFL   11,R,R,MICHIGAN/WRKDB.AF
$       FILE    12,X5S,12R
$       NOTE
$       NOTE    ************************************************************
$       NOTE    *  ACTIVITY 02 -- RUN THE RESTRUCTURER. ONOTE THAT         *
$       NOTE    *  THE $LIMITS CARD MUST BE ADJUSTED FOR EACH              *
$       NOTE    *  PARTICULAR RESTRUCTURING JOB.                           *
$       NOTE    ************************************************************
$       EXECUTE DUMP,NREST
$       LIMITS  40,65K,-5K
$       PRMFL   R*,R,S,MICHIGAN/RESTR.RS
$       FILE    H*,,40R
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       NOTE
$       NOTE    ------------------------------------
$       NOTE      THE FOLOWING SIX $FILE CARDS CORRESPOND
$       NOTE      TO THE SIX DATABASES COPIED IN ACTIVITY 01
$       NOTE
```

Figure 11-9

```
$       FILE    10,XOR              SOURCE RIF
$       FILE    11,X1S              TARGET RIF (SAVED FOR ACTIVITY 03)
$       FILE    12,X2R              'NAMES' TDL TABLES
$       FILE    13,X3R              'POINTERS' TDL TABLES
$       FILE    14,X4R              TARGET SDDL TABLES
$       FILE    15,X5R              DDL-WRITER WORK DATABASE
$       NOTE    ------------------------------------------
$       NOTE      THE FOLLOWING FOUR $FILE CARDS ARE USED
$       NOTE      IN THE PREPARATION OF THE TARGET RIF FOR
$       NOTE      RESTRUCTURING
$       NOTE
$       FILE    01,NULL             OUTPUT SINK (DDLA, DBINT)
$       FILE    02                  DDL TEXT FOR TARGET RIF
$       FILE    03                  DBTF FOR TARGET RIF
$       FILE    04                  USER HASH INPUT FILE TO DBINT
$       NOTE    ------------------------------------------
$       NOTE
$       DATA    05,,COPY
#TRANSLATION-NAME=NEW-PRESIDENTIAL-TO-PRESIDENTS-PARTS-STATES
#RUN=INITIALIZE
#EXECUTION-MONITOR=OFF
#ACTIVE-TDLAPS
ALL-TDLAPS-ACTIVE
#END-USER-INPUT
$       ENDCOPY
$       NOTE
$       NOTE    ********************************************************
$       NOTE    *  ACTIVITY 03 -- COPY THE NEW TARGET RIF DATABASE     *
$       NOTE    *  BACK TO THE PERMANENT FILE.  IF THE RESTRUCTURER    *
$       NOTE    *  ENCOUNTERED AN ERROR IN ACTIVITY 02, THE $IF CARD   *
$       NOTE    *  WILL PREVENT OVER-WRITING THE OLD TARGET DATABASE   *
$       NOTE    *  FILE WITH THE (POSSIBLY INCONSISTENT) NEW ONE.      *
$       NOTE    ********************************************************
$       IF      35,ENDJOB
$       UTILITY
$       FUTIL   01,02,RCOPY/1F/
$       FILE    01,X1R
$       PRMFL   02,N,R,MICHIGAN/PPS.TR
$       ENDJOB
```

Figure 11-9 (cont.)

```
$       IDENT
$       NOTE    ************************************************************
$       NOTE    *              EXECUTE THE WRITER JCL                      *
$       NOTE    *                                                          *
$       NOTE    *      UTILITY COPY OF TARGET RIF INTO TEMP. FILE          *
$       NOTE    ************************************************************
$       UTILITY
$       FUTIL   A1,A2,RCOPY/1F/
$       PRMFL   A1,R,R,MICHIGAN/PPS'.TR
$       FILE    A2,X1S,2R
$       NOTE    ************************************************************
$       NOTE    *      IF THIS IS THE FIRST RUN ON A TARGET                *
$       NOTE    *      DATABASE, THEN INCLUDE THIS ACTIVITY,               *
$       NOTE    *      ELSE SKIP DOWN TO THE COMPILATION                   *
$       NOTE    *                                                          *
$       NOTE    *        INITIALIZE TARGET IDS DATABASE                    *
$       NOTE    ************************************************************
$       PROGRAM OUTI
$       PRMFL   X1,REC,R,MICHIGAN/PPS'.ID
$       DATA    I*
IDS     INITIAL 1,12
$       NOTE    ************************************************************
$       NOTE    *      COMPILE THE MD SECTION INTO THE WRITER              *
$       NOTE    ************************************************************
$       IDS     NDECK
$       SELECTA MICHIGAN/COB1'.S
$       SELECTA MICHIGAN/PPS.MD
$       SELECTA MICHIGAN/COB2.S
$       NOTE    ************************************************************
$       NOTE    *              EXECUTE THE WRITER                          *
$       NOTE    ************************************************************
$       EXECUTE NREST
$       PRMFL   R*,R,S,MICHIGAN/WRITE.RS
$       FILE    H*,H1R,20R
$       LIMITS  5,50K,-2K,10000 USE APPROP. TIME & CORE REGMNTS
$       FILE    02,X1S          TARGET RIF
$       REMOTE  06              EXECUTION REPORT
$       REMOTE  07              ERROR REPORT
$       PRMFL   08,W,R,MICHIGAN/PPS.ST
$       DATA    15              USER INPUT PARAMETERS
STATES
YES     ALL
$       NOTE    ------------------------------------------------------------
$       NOTE    -      INCLUDE EITHER-
$       NOTE    -   $ FILE 16,X2R,2L   IF ALL RECORDS ARE WRITTEN
$       NOTE    -                              THIS RUN
$       NOTE    -            OR-
$       NOTE    -   $ PRMFL 16,W,S,BETWEEN RUNS SAVE FILE IF ONLY
$       NOTE    -                    PARTIAL DATABASE WRITE IS MADE
$       NOTE    ------------------------------------------------------------
$       FILE    16,X2R,2L
$       PRMFL   X1,REC,R,MICHIGAN/PPS'.ID
$       PRMFL   TR,R,R,MICHIGAN/TRANS.LR
$       NOTE    ************************************************************
$       NOTE    *      GO TO ENDJOB IF UNSUCCESSFUL WRITER RUN             *
$       NOTE    *          ELSE COPY UPDATED RIF BACK TO PRMFL             *
$       NOTE    ************************************************************
$       IF      /35,ENDJOB
$       UTILITY
$       FUTIL   A1,A2,RCOPY/1F/
$       FILE    A1,X1R
$       PRMFL   A2,W,R,MICHIGAN/PPS'.TR
```

Figure 11-10

APPENDIX A

IDS ANALYZER ERRORS

Documentation for all IDS Analyzer errors appears in this appendix. Each error message is given a number and cross-references are sometimes made. Information concerning the fatal/nonfatal nature of the error, the activity in which the error occurs, the cause, and the action to be taken to correct each error, are given.

Almost every error message is accompanied by the following output:

```
***ERROR...error message text

    CURRENT DATABASE IS = ****

    CURRENT 01 RECORD IS = ****

    CURRENT 02 FIELD IS = ****

    CURRENT VALIDATION IS = ****

    CURRENT 61 LEVEL IS = ****
```

By examining the values of the current database, record, etc. being processed, it is relatively easy for the user to examine the extended MD section to locate syntax and semantic errors.

Except for serious IDS Analyzer errors such as table overflow or exposed invalid program logic, no error is fatal to execution although occurrence of one syntax error may propagate several more. All user-correctable errors should be eliminated by appropriate modification to the extended MD section. The IDS Analyzer then should be rerun. There is no need to erase the previously built invalid SDDL tables file.

All error messages are sorted alphabetically to allow easy reference. The collating sequence observed is:

1) Numeric characters
2) Alphabetic characters
3) Special characters

1. "WARNING: UNABLE TO GENERATE USER SUPPLIED SYS-ENTRY POINT. PROBABLE CAUSE - NAME SUPPLIED IS NON-UNIQUE."

Cause: User has attempted to override IDS Analyzer's automatic system entry relation name generator by supplying a nonunique name for the relation in a 61 level SYS-ENTRY. This a nonfatal error in activity 3.

Action: If the user still wishes to rename the relation, the relation name given in the 61 level must be changed so that it is unique. No action is necessary if the name generated by the IDS Analyzer is acceptable to the user.

2. "***ABNORMAL TERMINATION IN IDS ANALYZER DUE TO ABOVE ERROR."

Cause: The IDS Analyzer has encountered an error from which it could not recover. This is a fatal error in activity 3.

Action: Analyze the error message directly above this and take the appropriate action.

3. "***FATAL ERROR..NON-ZERO RETURN CODE FROM ADBMS SYSTEM. CONTACT DATA TRANSLATION PROJECT."

Cause: The database management system supporting the IDS Analyzer has encountered an error in the Analyzer logic. This is a fatal error in activity 3.

Action: The user should not attempt to find the error. Contact the Data Translation Project.

4. "***ERROR IN 'ADHASL' - DUPLICATE DISPLACEMENTS."

Cause: Program logic error. It is fatal and occurs in activity 3.

Action: Contact the Data Translation Project.

5. "***ERROR IN 'ADHASL' - INVALID TYPE VALUE."

Cause: An IDS Analyzer support routine has been passed an invalid parameter. It is a fatal error which occurs in activity 3.

Action: Contact the Data Translation Project.

6. "***ERROR...01 RECORD-NAME NOT IN PARAMETER FILE"

Cause: The IDS Analyzer has encountered an 01 record in the IDS Query dictionary which was not defined by the user in the parameter file. It is a nonfatal error which occurs in activity 3.

Action: Insert the name of the 01 record into the parameter file and rerun the Analyzer.

7. "***ERROR...ADBMS ERROR IN ROUTINE BLDKEY AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. It is a fatal error which occurs in activity 4.

Action: Contact the Data Translation Project.

8. "***ERROR...ADBMS ERROR IN ROUTINE BLDVIR AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. It is a fatal error which occurs in activity 4.

Action: Contact the Data Translation Project.

9. "***ERROR...ADBMS ERROR IN ROUTINE CREVIR AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. It is a fatal error which occurs in activity 4.

Action: Contact the Data Translation Project.

10. "***ERROR...ADBMS ERROR IN ROUTINE ERRPK AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. It is a fatal error which occurs in activity 4.

Action: Contact the Data Translation Project.

11. "***ERROR...ADBMS ERROR IN ROUTINE INITDS AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. It is a fatal error which occurs in activity 4.

Action: Contact the Data Translation Project.

12. "***ERROR...ADBMS ERROR IN ROUTINE KEYACT AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. It is a fatal error which occurs in activity 4.

Action: Contact the Data Translation Project.

13. "***ERROR...ADBMS ERROR IN ROUTINE LNKEY AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. It is a fatal error which occurs in activity 4.

Action: Contact the Data Translation Project.

14. "***ERROR...ADBMS ERROR IN ROUTINE PROPV AT STMT-"

Cause: An Analyzer support routine has encountered an error while using its database management system, ADBMS. This fatal error occurs in activity 4.

Action: Contact the Data Translation Project.

15. "***ERROR...AN ITEM RECORD HAS NO NAME ON THE INAME SET."

Cause: An attempt to determine if an item specified by the user as a primary key of a record actually is an item in that record, has failed due to IDS Analyzer logic. This nonfatal error occurs in activity 3.

Action: Contact the Data Translation Project. SDDL tables generated by this run of the Analyzer should not be used.

16. "***ERROR...COLON EXPECTED AFTER NAME, NOT FOUND"

Cause: The syntax rules of the Analyzer called for a delimiting colon in a 61 level entry. This nonfatal error occurs in activity 3.

Action: Inspect the output immediately following the error message to determine at which 61 level the Analyzer was operating when the error occurred. Insert the colon in the proper place.

17. "***ERROR...COMMA EXPECTED AFTER NAME, NOT FOUND."

Cause: The syntax rules of the Analyzer called for a delimiting comma in a 61 level entry. This nonfatal error occurred in activity 3.

Action: Inspect the output immediately following the error message to determine at which 61 level the Analyzer was operating when the error occurred. Insert the comma in the proper place.

18. "***ERROR...CURRENTLY IMPOSSIBLE TO GENERATE A UNIQUE CONCAT-RELAY-NAME. CODE MUST BE MODIFIED."

Cause: The Analyzer was unable to generate a unique relation name for a contained-in-repeating group being modeled as a master/detail relation. This nonfatal error occurred during activity 3.

Action: Change the name of the contained-in-repeating group to one whose first twelve characters are unique. Rerun the IDS Analyzer.

19. "***ERROR...DB TYPE SPECIFIED IN PARAMETER FILE IS NOT ISP, IDS, OR SEQ"

Cause: The first line of the parameter file did not properly identify the database on which the Analyzer will work as being IDS, ISP, or Sequential. This fatal error occurred in activity 3.

Action: Check the first line of the parameter file to insure that it is in proper format.

20. "***ERROR...DEPENDENT KEY ITEM OF MATCH-KEY RELATION WAS NOT DEFINED."

Cause: The item which the user has specified as a dependent key in a match-key relation does not exist. This nonfatal error occurred in activity 3.

Action: Inspect the output immediately following the error message to determine at which 61 level the Analyzer detected the error. Make sure the dependent key item actually is an item in the dependent record and that it is spelled properly.

21. "***ERROR...DEPENDENT SPECIFIED IN MATCH-KEY RELATION IS UNDEFINED."

Cause: The record type which the user has specified as a dependent in a match-key relation does not exist. This nonfatal error occurred in activity 3.

Action: Inspect the output immediately following the error message to determine at which 61 level the Analyzer detected the error. Make sure the dependent record actually is a record in the augmented IDS MD section and that it is spelled properly.

22. "***ERROR...DUPLICATE ITEM DISPLACEMENTS WITHIN GROUP"

Cause: Program logic error. It is fatal and occurs in activity 3.

Action: Contact the Data Translation Project.

23. "***ERROR...ERRONEOUS DESCRIPTION."

Cause: A 61 level has been encountered by the Analyzer which it is unable to evaluate. This nonfatal error occurred in activity 3.

Action: Check the output following the error message to determine the 61 level which is at fault and the context in which it appears. Be certain that all the rules for coding 61 levels have been obeyed.

24. "***ERROR...EXPECTED NEXT DELIMITER IS '.' or ';'"

Cause: The syntax rules of the Analyzer called for a delimiting period or semi-colon in a 61 level entry. This nonfatal error occurred in activity 3.

Action: Inspect the output immediately following the error message to determine at which 61 level the Analyzer was operating when the error occurred. Insert the correct delimiter as specified in Sections 3.3-3.7.

25. "***ERROR...EXPECTED NEXT DELIMITER IS ',' OR ';' OR '.'"

Cause: The syntax rules of the Analyzer called for a delimiting comma, semi-colon, or period in a 61 level entry. This nonfatal error occurred in activity 3.

Action: Inspect the output immediately following the error message to determine at which 61 level the Analyzer was operating when the error occurred. Insert the correct delimiter as specified in Sections 3.3-3.7.

26. "***ERROR...EXPECTING GROUP NAME TO FOLLOW KEYWORD 'GROUP'. NOT FOUND."

Cause: A 61 level entry which was to supply the Analyzer with the name of the group whose primary key(s) would be specified next, had no group name in it. This nonfatal error occurred in activity 3.

Action: Using the output which immediately follows the error message, determine which 01 record the Analyzer was processing and in which 61 level the error occurred. Insert the proper group name after the keyword "GROUP:" or "G:".

27. "***ERROR...EXPECTING 'ITEM INFORMATION.' CLAUSE"

Cause: If a 61 level is coded beneath an item within the extended MD section, it must be OCCURS, EOG, DO-NOT-RESTRUCTURE, or ITEM-INFORMATION. The IDS Analyzer checks for ITEM-INFORMATION, hence this message will appear if one of the above clauses does not appear. This is a nonfatal error in activity 3.

Action: Refer to section 3.0 of the User Manual for correct rules on constructing level 61 entries. Rerun the IDS Analyzer.

28. "***ERROR...EXPECTING KEYWORD 'GROUP' OR 'G'. PROCESSING CONTINUES."

Cause: The preceding 61 level description alerted the Analyzer that information on a group's primary key(s) would follow in subsequent 61 levels. In this situation the Analyzer scans for the character string "GROUP" or "G". The error occurred because neither was found. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine where in the augmented IDS MD section the error occurred and make sure that the 61 level coding reflects the user's intent and follows IDS Analyzer rules.

29. "***ERROR...EXPECTING KEYWORD 'ITEMS' OR 'E-K-F' TO FOLLOW KEYWORD 'GROUP'. NOT FOUND."

Cause: The 61 level which immediately follows a 61 level description with the keyword 'GROUP' in it must contain either the keyword 'ITEMS' or 'I' or 'EXTERNAL-KEYS-FROM' or 'E-K-F'. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine where in the augmented IDS MD section the error occurred and correct the 61 level descriptions so that the Analyzer can extract the necessary information about the current group's primary key(s).

30. "***ERROR...EXPECTING KEYWORD 'RELATION'. NOT FOUND."

Cause: When the user is implementing phantom pointer or match-key relations through 61 level descriptions, Analyzer syntax rules require the user to supply a unique name for this relation. The name must be preceded by the keyword 'RELATION' or 'R'. This nonfatal error occurred in activity 3.

Action: Using the output which immediately follows the error message, determine where in the augmented IDS MD section the error occurred. Correct the 61 levels so that they conform to the syntax rules as described in Section 3.6.2 for phantom pointers, or Section 3.6.1 for match-key relations.

31. "***ERROR...EXPECTING 61 LEVEL DESCRIPTION RECORD"

Cause: An incomplete specification of all of the required level 61s was made for either match-key or phantom pointer relations. This nonfatal error occurred in activity 3.

Action: Refer to Section 3.3 for complete syntax for level 61s. Correct the error and rerun the IDS Analyzer.

32. "***ERROR...GROUP DECL'D TO HAVE KEYS ALONG RELATION IS NOT A MEMBER OF THAT RELATION. GROUP-_____ RELATION-_____"

Cause: The user has coded a 61 level description which has stated that a group could get 'external-keys-from' a relation in which it is not a member. This nonfatal error occurred in activity 4.

Action: Using the debug output in the error message, determine where in the augmented IDS MD section there appears the 61 levels that indicated this group could get external keys from the indicated relation. Check to be sure of which relation(s) the group is a detail and whether or not it actually needs external keys to identify itself uniquely.

33. "***ERROR...GROUP MUST BE A MEMBER OF AT LEAST ONE RELATION- "

Cause: Probably due to a previous error in executing the IDS Analyzer. By the time activity 4 is executed, the group specified must have been incompletely entered into the SDDL tables by not attaching at least one RELAY record along the RELMEM set (see Section 5.0). This nonfatal error occurred in activity 4.

Action: Correct all errors prior to this and rerun the IDS Analyzer.

34. "***ERROR..GROUP NOT UNIQUE"

Cause: The Analyzer has discovered an 01 level record name in the IDS Query dictionary which is not unique. This nonfatal error occurred in activity 3.

Action: The name of the 01 level record name which caused the error can be determined from the output which immediately follows the error message. Rename this 01 record and rerun the Analyzer.

35. "***ERROR...GROUP WHOSE FULL PRIMARY KEY CONSISTS OF SOME SET-SIG ITEMS MUST HAVE AN OWNER GROUP;- "

Cause: Error in specifying the EXTERNAL-KEYS-FROM information for the group specified. Relations used in that entry are not relations for which the group is a detail (member). This nonfatal error occurred in activity 4.

Action: Check extended MD section to ensure that all instances of EXTERNAL-KEYS-FROM entries abide by the rules of Section 3.5 of the User Manual. Correct errors and rerun the IDS Analyzer.

36. "***ERROR...GROUP-_____ DECLARED TO HAVE KEYS FROM RELATION-_____ DOES NOT HAVE ANY KEYS IN OWNER"

Cause: A group which had no primary keys of its own was declared in a 61 level entry to have external key(s) from a master group which also had no primary key(s). This nonfatal error occurred in activity 4.

Action: Using the debug output information in the error message, determine which group was the source of the error and recode the 61 level entries for that group making sure that it has a valid primary key or that its owner record does.

37. "***ERROR...GROUP-_____ DOESN'T HAVE ANY PRIMARY KEYS"

Cause: The group which is named in the error message was never indicated to have any primary keys or a source of primary keys in the 61 level coding for that group. This nonfatal error occurred in activity 4.

Action: Find this group in the augmented MD section and recode the 61 levels making sure to use one or both of the following formats: "61 ITEMS:" or "61 E-K-F:".

38. "***ERROR...GROUP-_____ HAS AN OWNER WITHOUT A PRIM. KEY"

Cause: Same as error #37.

Action: Same as error #37.

39. "***ERROR...GROUP-_____ MUST BE A MEMBER OF SOME RELATION. CAUSE DUE TO USER OR PREVIOUS IDS ANALYZER ERROR.

Cause: See error #33.

Action: See error #33.

40. "***ERROR...GROUP-_____ WAS DECLARED TO HAVE EXTERNAL KEYS ALONG-_____ BUT IS NOT A MEMBER OF THIS RELATION"

Cause: A 61 level "E-K-F:" description for the group named in the error message supplied the Analyzer with the erroneous information that the group was a member of the named set. This nonfatal error occurred in activity 4.

Action: From the debug output contained in the error message, determine where in the augmented MD section the invalid 61 level description occurred. Be sure to supply only relation names of which this group is a member.

41. "***ERROR...IDENT CAN'T BE SPECIFIED FOR IDS DATABASES."

Cause: The user has inadvertently used a 61 level description, "61 IDENT", which can be used only for ISP or sequential databases. This nonfatal error occurred in activity 3.

Action: Remove the 61 level which contains the "IDENT" description.

42. "***ERROR...IDENTIFICATION FOR RECORD TYPE NOT DEFINED"

Cause: For ISP and sequential databases, each 01 record must contain one item, denoted by the 61 IDENT: clause to be the record identifier. If this is not done, this nonfatal error occurs in activity 3.

Action: Define record identifiers using the rules of Section 3.7 of the User Manual via the 61 IDENT entry and rerun the IDS Analyzer.

43. "***ERROR...IDS ANALYZER GENERATED NAME (ITEM-NAME/IT) IS NOT UNIQUE, CHANGE NAME IN MD AND 61 LEVEL SECTIONS"

Cause: The Analyzer was unable to implement a repeating item as a concatenated relation because it could not assign unique names to the group and the item within it. This nonfatal error occurred in activity 3.

Action: As the error message suggests, the repeating item's name must be changed in the augmented MD section.

44. "***ERROR...ILLEGAL OR MISSING DELIMITER"

Cause: While parsing a 61 level description, the Analyzer was unable to find the delimiter that syntax rules would require given this 61 level description. This nonfatal error occurred in activity 3.

Action: From the output which follows the error message, determine which 61 level caused the error and insert the proper delimiter.

45. "***ERROR...ILLEGAL PAD CHARACTER"

Cause: The Analyzer was unable to find the new user-specified pad character in the 61 level description. This nonfatal error occurred in activity 3.

Action: From the output following the error message, determine which 61 level description containing the keyword 'PAD:' caused the error. Be sure that a new pad character is supplied following IDS Analyzer syntax rules.

46. "***ERROR...ILLEGAL SECTION NAME"

Cause: The Analyzer has encountered a 61 level description which it does not recognize. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine in which area of the augmented MD section the error occurred. Check to see that all the 61 levels in that area have valid syntax and are logically consistent with the Analyzer rules.

47. "***ERROR...ILLEGAL SYS-ENTRY"

Cause: An attempt to override the Analyzer's default system-owned relation name generator with the "SYS-ENTRY:" 61 level description has failed. The reason is either: 1) "SYS-ENTRY" syntax was violated; or 2) the record is not on a CALC or primary chain. This nonfatal error occurred in activity 3.

Action: Using the output which immediately follows the error message, determine which 61 level description was the source of the error. Check to see that all syntax rules were followed and that the record is a member of a CALC or primary chain.

48. "***ERROR...IMPOSSIBLE TO OPEN DB-____ RW-____IERR-____"

Cause: One of the two ADBMS databases used by the Analyzer cannot be opened for access. This is probably a control card error; databases must be random, the size must be at least 1 link. This fatal error occurred in activity 3.

Action: Contact the Data Translation Project if control cards are correct.

49. "***ERROR...IN MATCH-KEY RELATIONS, EXPECTING WORD-'KEY'"

Cause: The Analyzer has detected a syntax error while processing 61 levels describing a match-key relation. The next 61 level should have been: "61 KEYS:". This nonfatal error occurred in activity 3.

Action: Using the output following the error message, determine at which 61 level the Analyzer detected the error and insert the 61 level coding as described in Section 3.6.1.

50. "***ERROR...INVALID DEPENDENT NAME FOR PHANTOM RELATION"

Cause: While processing 61 level entries which describe a phantom pointer relation, the Analyzer was unable to find the name of the dependent in the relation. This nonfatal error occurred in activity 3.

Action: Using the output following the error message, determine at which 61 level the Analyzer detected the error and insert the proper 61 level coding as described in Section 3.6.2.

51. "***ERROR...INVALID NAME FOR PHANTOM RELATION."

Cause: While processing 61 level entries which describe a phantom pointer relation, the Analyzer was unable to find the name of the relation following the 61 level: "61 RELATION:". This nonfatal error occurred in activity 3.

Action: Same as error #50.

52. "***ERROR...INVALID POINTER NAME FOR PHANTOM RELATION"

Cause: While processing 61 level entries which describe a phantom pointer relation, the Analyzer was unable to find the name of the item to be used as a pointer following the 61 level: "61 POINTER:". This nonfatal error occurred in activity 3.

Action: Same as error #50.

53. "***ERROR...INVALID 61 LEVEL DESCRIPTION."

Cause: While processing 61 level descriptions which followed "02 TRANSLATION INFORMATION SIZE 0", the Analyzer encountered a 61 level which is not correct given the context in which it appears. This nonfatal error occurs in activity 3.

Action: Using the output following it, determine which 61 level generated the error message. It is possible that the error was initiated by the 61 level coding immediately preceding the current 61 level. Therefore, make sure that the last few 61 levels reflect the user's intent and follow Analyzer syntax rules.

54. "***ERROR...ITEM WITH DBKEY (SEE DUMP REPORT) OF-____ DECL'D TO BE PRIMARY KEY MORE THAN ONCE MULTIPLE DECLARATIONS IGNORED"

Cause: The user has specified an item as a primary key of a record more than once in the 61 level description, "61 PRIMARY-KEYS:". This nonfatal error occurred in activity 4.

Action: None necessary.

55. "***ERROR...KEYWORD 'DEPENDENT' EXPECTED"

Cause: While processing 61 levels describing match-key or phantom pointer relations, the Analyzer expected the 61 level "61 DEPENDENT:" to appear next but did not find it. This nonfatal error occurred in activity 3.

Action: Using the output following the error message, determine at which 61 level the error occurred and insert the correct 61 level description as described in Section 3.6.2 for phantom pointers, or Section 3.6.1 for match-key relations.

56. "***ERROR...KEYWORD 'PARENT' NOT FOUND"

Cause: While processing 61 levels describing match-key relations, the Analyzer expected the 61 level "61 PARENT:" to appear next but did not find it. This nonfatal error occurred in activity 3.

Action: Using the output following the error message, determine at which 61 level the error occurred and insert the correct 61 level description as described in Section 3.6.1 for match-key relations.

57. "***ERROR...LENGTH OF NAME OVER 30 CHARACTERS"

Cause: The user has specified a group, item, or relation name which is greater than 30 characters long in the augmented IDS MD section. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine where in the MD section the error occurred and change the name to one less than or equal to 30 characters in length.

58. "***ERROR...LOOPS OF PRIMARY KEYS MUST HAVE BEEN DEFINED BY THE USER. CHECK TO SEE THAT PRIMARY KEYS FOR ANY GROUP AREN'T MADE UP OF PRIMARY KEYS THAT THE GROUP GENERATES TO ITS MEMBERS. APPLY THIS RECURSIVELY."

Cause: A basic IDS Analyzer rule has been violated. A circular path, describing where groups get primary keys has been defined in the 61 levels. This nonfatal error occurred in activity 4.

Action: Refer to the debugging report which follows the error message. 61 levels which propagated the error must be recoded.

59. "***ERROR...MATCH-KEY ITEM NOT DEFINED IN PARENT"

Cause: The item specified by the user as a match-key item in the parent of the relation is not an item which appears in the parent. This nonfatal error occurred in activity 3.

tion: Recode the 61 level entry "61 KEYS:", so that the key item specified is a valid item within the parent. The 61 level which caused the error can be determined from the output which follows the error message.

60. "***ERROR...PHANTOM RELATIONS CANNOT BE USED FOR ISP OR SEQ DATABASES"

Cause: The user has attempted to implement a phantom pointer relation in an ISP or sequential database. This nonfatal error occurred in activity 3.

Action: Because this is prohibited by the IDS Analyzer, the user should remove the 61 level coding relevant to the error and implement the desired relationship in another acceptable way.

61. "***ERROR...MISPLACED EOG. STATEMENT."

Cause: The Analyzer has detected a 61 level 'END OF GROUP' description which is out of context based on the preceding 61 level descriptions. This nonfatal error occurred in activity 3.

Action: Inspect the output which follows the error message to determine which 61 level contains the misplaced EOG.if it is extraneous, or rewrite the 61 level descriptions to follow IDS Analyzer syntax rules as described in Section 3.3.

62. "***ERROR...MISSING PRIME"

Cause: Correct 61 level syntax in this situation caused the Analyzer to look for a prime ('). This nonfatal error occurred in activity 3.

Action: From the output which follows the error message and the intent of the user in coding it, determine which 61 level caused the error. Refer to the syntax rules covering this situation to determine where the prime is to be placed.

63. "***ERROR...MORE THAN 5 DATA BASES ENTERED IN PARAM. FILE."

Cause: The user specified more than five database names in the parameter file. This nonfatal error occurred in activity 3.

Action: The IDS Analyzer restricts the number of databases on which it will work to five. In order to use the Analyzer module, the user should observe this limitation.

64. "***ERROR...NAME IN CLAUSE MUST BEGIN WITH LETTER OR DIGIT"

Cause: User name must begin with a letter. Constants must begin and consist entirely of digits. Violation results in this nonfatal error which occurs in activity 3.

Action: Rewrite offending user name or constant and rerun the IDS Analyzer.

65. "***ERROR...NAME OF RELATION IS NOT UNIQUE"

Cause: The name of the relation supplied by the user to be used in a match-key relation is not unique. This nonfatal error occurred in activity 3.

Action: Using the output following the error message, determine the location of the 61 level containing the non-unique relation name. Change the relation name.

66. "***ERROR...NO DATA BASES SPECIFIED IN PARAM. FILE"

Cause: After reading the entire parameter file, the Analyzer was unable to find the name of any database. This fatal error occurred in activity 3.

Action: Check the parameter file to be sure that the name of the user's database is specified according to the rules stated in Section 5.6.

67. "***ERROR...NO DELIMITER FOUND"

Cause: The Analyzer expected to find a legal delimiter but could not. This nonfatal error occurred in activity 3.

Action: Inspect the output following the error message to determine in which 61 level the Analyzer expected to find a delimiter. Make sure the syntax is consistent with the user's intent and the IDS Analyzer syntax rules.

68. "***ERROR...NO GROUPS DEFINED IN DATABASE"

Cause: The IDS Analyzer failed to generate any groups in the SDDL tables. This nonfatal error occurred in activity 4.

Action: This error will not occur without other errors preceding it. Correcting these errors should eliminate this one.

69. "***ERROR...NO ITEM RECORDS ARE PRESENT ON THE ITEM SET IN THE SDDL TABLES FOR THE CURRENT GROUP"

Cause: While attempting to determine if an item specified by the user as a key of a group is actually an item within that group, the Analyzer discovered that the group has no items. This nonfatal error occurred in activity 3.

Action: This error can occur only as a result of prior errors. If, after correcting all prior errors, this message still appears, contact the Data Translation Project.

70. No longer implemented

71. "***ERROR...NUMBER OF MATCH-KEYS IN PARENT NOT = TO NUMBER OF MATCH-KEYS IN DEPENDENT"

Cause: In defining a match-key relation in the 61 levels, the user mismatched the number of match-keys in the parent and the dependent. This nonfatal error occurred in activity 3.

Action: Determine where the match-key relation is defined in the augmented MD section by examining the output following the error message. Change the 61 level coding so that the numbers of match-keys in the parent and the dependent are equal.

72. "***ERROR...NUMBER OF OCCURRENCES IS ZERO"

Cause: User specified OCCURS 0 TIMES in level 61 entry. This nonfatal error occurred in activity 3.

Action: Groups must occur at least once. Rewrite the level 61 entry according to rules in Section 3.3 and rerun the IDS Analyzer.

73. "***ERROR...PARENT OF MATCH-KEY RELATION IS NOT DEFINED"

Cause: The name of the group which the user specified as the parent in a match-key relation does not exist. This nonfatal error occurred in activity 3.

Action: Using the output following the error message, determine where the Analyzer was in the augmented IDS MD Section when the error was detected. Replace the name of the parent group with a valid one in accordance with the rules set forth in Section 3.6.1.

74. "***[illegible]NTOM RELATION NAME NOT UNIQUE"

Cause: The name of the phantom relation supplied by the user is not unique to this augmented MD section. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine which 61 level contains the nonunique relation name and change it.

75. "***ERROR...'POINTER' NOT FOUND"

Cause: The syntax rules for coding 61 levels to define phantom pointer relations require the key word 'POINTER' which the Analyzer was unable to find. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine where in the MD section the phantom pointer relation was defined. Make sure that the syntax rules of Section 3.6.2 are obeyed.

76. "***ERROR...RELATION-_____ DOESN'T HAVE A MEMBER GROUP"

Cause: Previous error in IDS Analyzer. Detail (member) of either a chain, match-key, or phantom relation does not exist. This nonfatal error occurred in activity 4.

Action: Correct all previous errors and rerun IDS Analyzer.

77. "***ERROR...SIZE CONFLICT BETWEEN LENGTH OF GROUP AND LENGTHS OF ELEMENTARY ITEMS"

Cause: A group of size "x" has been defined in the IDS MD section, but the sum of its elementary items is not equal to its declared size. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine where in the augmented IDS MD section the error occurred. Following the rules set forth in Section 3.2.1, recalculate the length of the group.

78. "***ERROR...SYMBOL DEFINED IN 61 LEVELS IS NOT IN THE IDS MD SECTION, SYMBOL IS: "

Cause: The user specified a match-key item, phantom pointer item, or dependent group name within level 61 entries for which there is no occurrence within the IDS MD section. The most likely cause is misspelling. This nonfatal error occurred in activity 3.

Action: Correct erroneous level 61 entry where the "symbol" appears and rerun the IDS Analyzer.

79. "***ERROR...SYNTAX OF IDENT CLAUSE IS INCORRECT"

Cause: The Analyzer was unable to successfully parse through the 61 level description being used to define an IDENT item for the current record. This nonfatal error occurred in activity 3.

Action: Using the output which follows the error message, determine at which 61 level the error occurred and rewrite it to follow correct syntax rules.

80. "***ERROR...THERE IS NO NEXT POINTER FOR MD"

Cause: Query dictionary was incorrectly built by IDS Translator in query mode. Each Master Definition record has a field within it containing the displacement of the next pointer for the chain. If this field is zero, the above nonfatal error occurs in activity 3.

Action: Check extended IDS MD section closely to ensure that all 98 chain CHAIN MASTER entries are correctly written. Refer to the IDS Programmer's Guide for complete syntax rules.

81. "***ERROR...TOO MANY 02 LEVELS MAX = 50"

Cause: Internal tables in the IDS Analyzer can handle only 50 02 levels per 01 record. This fatal error occurred in activity 2.

Action: Change the MD section to have fewer than 50 02 levels if it is a target MD; otherwise, contact the Data Translation Project.

82. "***ERROR...UNABLE TO SET CURRENCY OF HASPK AND EXKEYS SETS. NO GRP RECORD TO MATCH USER SUPPLIED GROUP NAME."

Cause: This error can occur only as a result of previous errors or program logic faults. It is a nonfatal error which occurs in activity 3.

Action: If, after all previous errors have been corrected, this error still remains, contact the Data Translation Project.

83. "***ERROR...USER CODED MULTIPLE TRANSLATION INFORMATION'S. 61 LEVELS IGNORED EXCEPT UNDER 1ST OCCURRENCE"

Cause: The 02 level, "02 TRANSLATION-INFORMATION SIZE 0" has occurred more than once under the same 01 record. This nonfatal error occurs in activity 3.

Action: Remove the redundant "02 TRANSLATION-INFORMATION SIZE 0". Its location can be determined by examining the output which follows the error message.

84. "***ERROR...USER DEFINED MORE THAN 75 GROUPS, VECTORS FOR BLDKY MUST BE REDIMENSIONED."

Cause: An internal Analyzer table overflowed because the user specified more than 75 groups in the augmented MD section. This fatal error occurred in activity 4.

Action: Although this error will not terminate execution, it probably will propagate further errors. Contact the Data Translation Project if the problem occurred in a source MD section. Otherwise, reduce the MD to less than 75 groups, if possible.

85. "***ERROR...USER HAS SPECIFIED AN INVALID GROUP NAME."

Cause: The name of the group supplied by the user in the "Primary-Keys" section does not exist. This nonfatal error occurred in activity 3.

Action: Determine which 61 level description contained the invalid group name by inspecting the output which follows the error message. Replace the incorrect name with one which is consistent with the rules stated in Section 3.5.

86. "***ERROR...USER HAS SPECIFIED AN INVALID ITEM AS A PRIMARY KEY"

Cause: The item named by the user to be a primary key of a group is not an item belonging to that group. This nonfatal error occurred in activity 3.

Action: Determine from the output which follows the error message which 61 level description contains the source of the error. Delete the invalid item's name and make sure that all other items declared as keys for the group actually are items within that group.

87. "***ERROR...USER SYNTAX IS WRONG"

Cause: The Analyzer is unable to interpret the current 61 level given the context in which it appears. This nonfatal error occurred in activity 3.

Action: Determine from the output that follows the error message which 61 level generated the error message and check that 61 level, as well as several others before it, to make sure that all relevent syntax rules are obeyed.

88. "***ERROR...*****DATABASE FOR SDDL TABLES IS TOO SMALL, ACTIVITY TERMINATED."

Cause: This error message is preceded by error message #9. The error occurred because the Analyzer's supporting database management system, ADBMS, has run out of room in the database which holds the SDDL tables. This is ADBMS error #17. It is a fatal error which occurred in activity 4.

Action: Recreate the SDDL tables database to a larger size and rerun the Analyzer.

89. "***WARNING...SET-SIG ITEM NAME FOR-_____ TRUNCATED ON RIGHT TO 240 CHARS"

Cause: A set-significant item name was generated with a length greater than 240 characters, the upper limit allowed. Truncation occurred to trim it to 240 characters. This nonfatal error occurred in activity 4.

Action: No action is necessary. This message is generated simply to alert the user to the fact that some set-significant item names may have an unusual appearance due to the truncation.

90. "*******ERROR IN NAMFXS*******"

Cause: An Analyzer support routine was unable to generate a needed name. The user specified more than twenty items within one record whose first six characters are identical or there are more than twenty record or relations whose names have identical first six characters. This nonfatal error occurred in activities 3 and 4.

Action: Change user names so that first six characters of records, relations, and items are not identical.

APPENDIX B

TDL ANALYZER ERRORS

Activity 1 Errors

Activity 1 is an execution of the system UTILITY routines. A description of the errors generated can be found in the UTILITY manual.

Activity 2 Errors

1. ANALYZER ERROR, STACK OVERFLOW

Cause: The error recovery procedure was not able to recover properly.

Action: Correct any previous errors.

2. MACRO LITERAL TRUNCATED TO ONE LINE

Cause: The literal portion of a macro declaration exceeded one line. This is a warning.

Action: Make sure that the macro literal begins and ends on the same line. If the macro literal is more than one line long, then several literals may be needed.

3. MACRO-NAME INSIDE OF MACRO, IGNORED

Cause: A macro name was found inside the macro literal. This is a warning.

Action: Remove imbedded macro names.

4. SYNTAX ERROR, ILLEGAL TOKEN PAIR

Cause: A user syntax error. This is a nonfatal error.

Action: Remove the syntax error.

5. SYNTAX ERROR, NO PRODUCTION MATCHES

Cause: A nonfatal user syntax error.

Action: Remove the syntax error.

6. ERROR RECOVERY PROCEDURE BEGINNING

Cause: A previous syntax error has caused the error recovery procedure to begin. This is a warning.

Action: Remove the previous syntax error.

7. ERROR RECOVERY PROCEDURE COMPLETED

Cause: A previous syntax error initiated the error recovery procedure which is now complete. Lines between the "beginning" and the "completed" error recovery procedure warning messages were not processed. This is a warning.

Action: Remove the previous syntax error.

8. EOF FOUND, SECTION COMPILING STOPPED

Cause: A previous syntax error initiated the error recovery procedure which could not recover before the end-of-file was reached. This is a nonfatal error.

Action: Remove the previous error.

9. ANALYZER ERROR, R-W TABLE OVERFLOWED

Cause: A fatal error in the TDL Analyzer.

Action: Ensure that the TDL parse tables are connected to the correct I/O unit.

10. ILLEGAL R.W. - ENTERED AS USER NAME

Cause: This nonfatal error should appear only in future implementations of the TDL Analyzer.

Action: Contact the Data Translation Project.

11. ANALYZER ERROR, SYM TABLE OVERFLOW

Cause: Too many user names and macros were entered; the symbol table has overflowed. This is a fatal error.

Action: Recompile the TDL Analyzer with a larger BLOCK DATA.

12. MACRO MUST BE FOLLOWED BY A NAME

Cause: The macro name is missing in the macro definition. This is a nonfatal error.

Action: Correct the macro definition syntax error.

13. NAME PREVIOUSLY USED - MACRO IGNORED

Cause: The macro name was previously used. This is a nonfatal error.

Action: Select a new macro name.

14. LITERALLY SHOULD FOLLOW MACRO-NAME

Cause: Improper macro definition syntax. This is a warning.

Action: Correct the syntax error.

15. A LITERAL MUST FOLLOW "LITERALLY"

Cause: Improper macro definition syntax. This is a nonfatal error.

Action: Correct the syntax error.

16. SEVERE ERROR(S), MACRO NOT ENTERED

Cause: Previous errors have caused the macro definition to be ignored. This is a nonfatal error.

Action: Correct the previous errors.

17. ANALYZER ERROR, FREE AREA OVERFLOWED

Cause: Too many user names and macro names were entered. This is a fatal error.

Action: Recompile the TDL Analyzer with a larger block data.

18. ANALYZER ERROR, FREWRD OVERFLOWED

Cause: Too many user names and macro names were entered. This is a fatal error.

Action: Recompile the TDL Analyzer with a larger block data.

19. ILLEGAL CHARACTERS IN INPUT, IGNORED

Cause: Illegal characters appeared in the input stream. This is a warning.

Action: Remove the illegal characters.

20. NAMELENGTH > 256, WILL BE TRUNCATED

Cause: The user name is more than 256 characters long. This is a warning.

Action: Change the user name.

21. INTEGER OVERFLOW, VALUE TRUNCATION

Cause: The integer has exceeded its maximum value. This is a warning.

Action: Change the integer.

22. LITERAL LENGTH > 256, WAS TRUNCATED

Cause: The literal is more than 256 characters long. This is a warning.

Action: Change the literal.

23. LITERAL DOES NOT END WITH A PRIME

Cause: The closing prime for the literal is missing. This is a warning.

Action: Correct the syntax.

24. COMMENT DOES NOT TERMINATE WITH A */

Cause: A comment was not enclosed by /* and */. This is a warning.

Action: Correct the syntax error.

25. EOF MISSING, EOF CARD INSERTED

Cause: The last card processed was not an EOF card. This is a warning.

Action: Insert an EOF card after the last card.

26. TARGET RECORD NOT FOUND

Cause: The target record with the specific name does not exist. This is a nonfatal error.

Action: Check the target SDDL table dump (part of the IDS Analyzer output) to ensure that the target record name used in the TDL description is correct.

27. TDLAP NAME TRUNCATED TO 12 CHARACTERS

Cause: A TDLAP name is longer than 12 characters. This is a warning.

Action: Select a shorter TDLAP name.

28. TARGET ITEM ASSIGNED MORE THAN ONCE

Cause: A target item has been assigned more than once in the same TDLAP. This is a warning.

Action: Determine which assignments, if any, are erroneous and remove them.

29. TARGET ITEM NOT FOUND

Cause: The target item does not exist in the specified target record. This is a nonfatal error.

Action: Check the target SDDL table dump (part of the IDS Analyzer output) to ensure that the target item name used in the TDL description is correct.

30. SOURCE RECORD NOT FOUND

Cause: The source record does not exist. This is a nonfatal error.

Action: Check the source SDDL table dump (part of the IDS Analyzer output) to ensure that the source record name used in the TDL description is correct.

31. ID BUFFER OVERFLOW

Cause: More than 50 ID names were specified on a single TDLAP. This is a nonfatal error.

Action: Remove unneeded ID names. If more than 50 ID names are needed, recompile the TDL Analyzer with a larger block data.

32. ACCESS VIA RELATION NOT FOUND

Cause: The access via relation does not exist. This is a nonfatal error.

Action: Check the source SDDL table dump (part of the IDS Analyzer output) to ensure that the access via relation name in the TDL description is correct.

33. SET SIG ITEMS CANNOT BE ASSIGNED

Cause: A source set-significant item was used in a TDLAP where ACCEPT IF NULL was specified for one or more of the source records. This is a nonfatal error.

Action: Alter the TDLAP so that all assignments and qualifications refer only to actual items.

34. SOURCE ITEM NOT FOUND

Cause: The source item does not exist in the specified source record. This is a nonfatal error.

Action: Check the source SDDL table dump (part of the IDS Analyzer output) to ensure that the source item name in the TDL description is correct.

35. QUALIFICATION LINK NAME TRUNCATED

Cause: The qualification link name is more than six characters long. This is a nonfatal error.

Action: Correct the link name.

36. INVALID ID FOR RECORD

Cause: The specified ID does not match the record given. This is a nonfatal error.

Action: Check the source record and ID to determine which is in error.

37. FROM SOURCE RECORD NOT FOUND

Cause: The "from" source record has not yet been placed on the access tree (TDLAP). This is a nonfatal error.

Action: Check to see that the correct record name and relation name were used. Then make sure that the "from" source record was previously placed on the DTLAP. The ordering of source records may need to be altered.

38. CONVERSION LINK NAME TRUNCATED

Cause: The conversion link name is more than 6 characters long. This is a nonfatal error.

Action: Use the correct link name.

39. NULL VAL ILLEGAL W/O ACCEPT IF NULL

Cause: A null value was specified for a target item and Reject IF NULL was specified (or defaulted) for the source record.

Action: Determine whether the source record should be ACCEPTed or REJECTed if null. If ACCEPT IF NULL, this adds this clause to the source record statement. If REJECT IF NULL, then remove the NULL VALUE clause(s).

40. NO ACTUAL DATA TO ASSIGN

Cause: The user specified ACTUAL DATA IN ORDER when no actual data exists. This is a warning.

Action: None required; however, be aware of the fact that no actual data has been assigned.

41. ACTUAL DATA TOO BIG FOR TARGET RECORD

Cause: The user specifies ACTUAL DATA IN ORDER for a source record which contains more data than will fit in that target record. This is a nonfatal error.

Action: Use explicit item assignments instead of ACTUAL DATA IN ORDER.

42. SET SIG ITEMS CANNOT BE ASSIGNED

Cause: Set-significant source items were used on a TDLAP where ACCEPT IF NULL was specified for one of the source records. This is a nonfatal error.

Action: Alter the TDLAP so that all assignments and qualifications are based on actual source items.

43. CONFLICTING SOURCE RECORD AND ACCESS

Cause: The source record is not a member or an owner of the access via relation. This is a nonfatal error.

Action: Check the source SDDL table dump (part of the IDS Analyzer output) to ensure that the source record name in the TDL description is correct.

44. OWNER/MEMBER OR MEM/OWN MUST BE USED

Cause: The source record is both the owner and the member of the relation. This is a nonfatal error.

Action: Determine if the record should be the owner or member of the relation and insert an OWNER/MEMBER or a MEMBER/OWNER clause.

45. NO NODE TO ATTACH SOURCE RECORD TO

Cause: The source record at the other end of the access via relation is not a node in the current TDLAP. This is a nonfatal error.

Action: Ensure that the correct access via relation was used and that the TDLAP tree nodes are constructed from root to leaves.

46. OWNER/MEMBER OR MEM/OWN INCORRECT

Cause: Owner/member was specified when member/owner is actually correct, or vice versa. This is a nonfatal error.

Action: Ensure that source record and access via relations are correct and change owner/member to member/owner or vice versa.

47. NODE ALREADY EXISTS ON ACCESS TREE

Cause: A source record has been placed on the TDLAP more than once. This is a nonfatal error.

Action: Ensure that duplicate nodes on the TDLAP are desired.

48. ID MANDATORY FOR THIS SOURCE RECORD

Cause: The source record needs an ID to uniquely identify it. This is a nonfatal error.

Action: Add an ID = clause to the source record.

49. ID AND SOURCE RECORD NAME CONFLICT

Cause: The source record with the ID specified does not have the same name as the source record name specified. This is a nonfatal error.

Action: Determine which source record is desired and change the ID and/or source record specified.

50. QUALIFICATION ITEM NOT FOUND

Cause: The qualification item does not exist in the specified source group. This is a nonfatal error.

Action: Determine the source record where the qualification item exists and add or correct the From clause.

51. TDL DATA BASE TOO SMALL or PREVIOUS ERRORS OR ERRORS IN THE SDDL TABLES HAVE CAUSED A FATAL ERROR.

Cause: Either the TDL database file is not large enough or an unreasonable error has occurred.

Action: Recreate the TDL tables with a larger size. Check the IDS Analyzer output to ensure that no errors occurred and correct any errors encountered in the partial run of the TDL Analyzer.

Activity 3 Errors

ERROR ENCOUNTERED WHILE SETTING SYSACCS

Action: Contact the Data Translation Project.

Activity 4 Errors

ERROR IN CONSTRUCTION OF COMPATIBILITY SETS

Action: Contact the Data Translation Project.

Activity 5 Errors

1. TDL TABLE INCONSISTENCY OR TDL DUMP PROGRAM ERROR

Action: This is a fatal error. Any errors found in activity 2 should be corrected. If no activity 2 errors were detected, then contact the Data Translation Project.

APPENDIX C

READER ERRORS

These error messages are found on report code 06 Activity 2 (Activity 3 for the IDS Reader).

1. ###ERROR.....ACCIDS HAS AN IDS ERROR.

Cause: An IDS error (not abort) has occurred in the Accessor module.

Action: Make sure the proper MD section was compiled with the Accessor. Check the IDS compilation for errors.

2. ###ERROR.....ACCISP HAS AN ISP ERROR.

Cause: An ISP error has occurred in the Accessor module.

Action: Be sure the ISP database is in the proper order. An error explanation will probably be in the execution report ($$).

3. ###ERROR.....ADBMS ERROR IN FILLIT.

Cause: An error condition has arisen in the internal database management system.

Action: This error message will precede an ADBMS error message and trace (see Appendix F for an explanation of ADBMS errors). If the error is #17 and the "DB=2" then increase the size of the source RIF and rerun the Reader. If the error is #17 and the "DB=3", then increase the size of DRT file and rerun the Reader. Otherwise, this error can be corrected only by Data Translation personnel.

4. ###ERROR.....CALL TO NXTITM, NO CURRENT GROUP.

Cause: There is no current group in the SDDL tables.

Action: This indicates a serious error has occurred in the logic of the Reader. It will probably be preceded by other errors which should be corrected first. Otherwise, it can be corrected only by Data Translation personnel.

5. ###ERROR.....COULDN'T GET ITEM INTO BUFFER, name.

Cause: An error has occurred while moving data from the source database to an internal buffer. This is a fatal error.

Action: Can be corrected only by Data Translation personnel.

6. ###ERROR.....CR ERROR# n

Cause: An ADBMS error has occurred while creating a record in the SRIF.

Action: Same as #3.

7. ###ERROR.....DATABASE NAME NOT FOUND. name IGNORED.

Cause: The database name, "name", which the user specified as a run-time parameter could not be found in the SDDL tables.

Action: Put the proper database name in the "NAME=" parameter in the run-time parameter file and rerun the Reader. The correct name can be found in IDS Analyzer ouptut. Source databases which have already been read in need not be reprocessed.

8. ###ERROR IN ADBMS DDL ANALYZER.

Cause: This error message will be printed after other errors have occurred.

Action: If possible, correct all errors which preceded this error. Otherwise, this can be fixed only by Data Translation personnel.

9. ###ERROR IN ADBMS DB INITIALIZATION.

Cause: See #8.

Action: See #8

10. ###ERROR IN DDL WRITER.

Cause: An error occurred while writing the ADBMS DDL for the SRIF.

Action: Correct all errors which preceded this message. The user should be sure that the IDS Analyzer run creating the Source SDDL tables had no errors.

11. ##ERROR.....PROBLEMS IN GETCRG.

Cause: The Source SDDL tables were created incorrectly.

Action: Recheck IDS Analyzer output, recreate the Source SDDL tables, and then run the Reader.

12. ###ERROR.....RECORD TYPE n NOT FOUND, SKIPPED.

Cause: The IDS record type n is not described in the source SDDL tables.

Action: Include the MD section for record type n with the rest of the MD section and run the IDS Analyzer to create the source SDDL tables. Then run the Reader.

13. ###ERROR PROBLEMS WITH MVB IN "STRPOP".
###FATAL ERROR, RIF WILL BE WRONG.

Cause: See #5.

Action: See #5.

14. ###ERROR.....TRUNCATED DOUBLE WORD INTEGER. SET TO ZERO.
OCTAL OF BUFFER (1) AND BUFFER (2).

Cause: An error has occurred while converting a double word integer to a single word integer.

Action: This problem cannot be corrected. Due to data representation restrictions in ADBMS, all double word integers must be converted to single word integers. If truncation occurs in the conversion subroutine, that data item is set to zero. The line following this error message will have the octal representation of the double word integer which now equals zero.

15. ###FATAL ERROR...AMS ERROR# n.

Cause: See #3.

Action: See #3.

16. ###FATAL ERROR...POPULATOR NOT OPENED.

Cause: The DRT was not properly initialized.

Action: This error message should be preceded by other errors which should be corrected.

17. ###FATAL ERROR...PROBLEMS WITH MVB IN MAIN.

Cause: See #5.

Action: See #5.

18. ###WARNING...THE SET name1 IN THE RECORD name2 IS ASSUMED TO BE A PHANTOM CHAIN.

Cause: The "NEXT" pointer field for the master record of set name1 does not exist.

Action: Note that this message will be printed once for every occurrence of name1 and name2. If the set name1 is not a phantom pointer relation, the 61 levels should be corrected and the IDS Analyzer should be rerun. If the 61 levels are correct and the set name1 is not a phantom pointer relation, an error has occurred in one of the sub-modules and it can be fixed only by Data Translation personnel.

19. ***ERR IN GETITM--ARRAY DIMENSION OF PITEM & DITEM IS TOO SMALL.

Cause: Too many match-key items in one match-key relation.

Action: Can be fixed only by Data Translation personnel.

20. ***ERR IN GETITM--STATEMENT# n

Cause: This error message is printed only after other errors have occurred.

Action: If possible, correct all other errors which precede this message. Otherwise, this error can be fixed only by Data Translation personnel.

21. ***ERR IN MKYREL--ERROR RETURN FROM GETITM.

Cause: See #19.

Action: See #19.

22. ***ERR IN MKYREL--STATEMENT#n.

Cause: See #20.

Action: See #20.

23. ***ERROR IN RT2SNM.

Cause: See #5.

Action: See #5.

Note: Errors 24 through 33 are all the same.

24. ...POPULATOR - CHKSET GETS ADBMS ERROR# n
25. ...POPULATOR - CLSPOP GETS ADBMS ERROR# n
26. ...POPULATOR - CPYSET GETS ADBMS ERROR# n
27. ...POPULATOR - FANCHN GETS ADBMS ERROR# n
28. ...POPULATOR - GETFST GETS ADBMS ERROR# n
29. ...POPULATOR - GNEXT GETS ADBMS ERROR#n
30. ...POPULATOR - INSFST GETS ADBMS ERROR# n
31. ...POPULATOR - INSNXT GETS ADBMS ERROR# n
32. ...POPULATOR - POPINT GETS ADBMS ERROR# n
33. ...POPULATOR - UNION GETS ADBMS ERROR# n

Cause: See #3.

Action: See #3.

34. ...POPULATOR COPYING ERROR Key1 Key2.

Cause: The Populator was unable to move all the member records owned by Key1 to Key2.

Action: Unless this error was preceded by some other errors, it can be corrected only by Data Translation personnel.

35. ...POPULATOR - DETAIL RECORD POINTS TO ITSELF.

Cause: The Populator tried to link a member record into a set. The record's "next" field was equal to its reference code.

Action: The pointer fields are incorrect in the source IDS database. They must be corrected before the Reader will run. Data Translation personnel may be required to help find the record instances in the IDS database which have the problem.

36. ...POPULATOR - ERROR FROM GFIRST.

Cause: This error message will be preceded by other errors.

Action: If possible, follow the suggested action for all errors which preceded this message. Otherwise, this error can be corrected only by Data Translation personnel.

37. ...POPULATOR - ERROR FROM INSFST.

Cause: See #36.

Action: See #36.

38. ...POPULATOR - ERROR FROM UNION.

Cause: See #36.

Action: See #36.

39. ...POPULATOR FOUND TWO REAL PARENTS IN IDS DB.

Cause: Two records were found to be the owner of the same set instance.

Action: See #35.

40. ...POPULATOR - INSFST INTERNAL ERROR

Cause: See #39.

Action: See #35.

41. ...POPULATOR - INSNXT INTERNAL ERROR.

Cause: See #39.

Action: See #35.

42. ...POPULATOR - INTERNAL ERROR IN DRT.

Cause: A relation is missing the DRT or a sibling name has been omitted.

Action: This error can be corrected only by Data Translation personnel.

43. ...RECORD TYPE IS NEITHER MASTER NOR DETAIL.

Cause: The Populator module was passed bad data.

Action: This error can be corrected only by Data Translation personnel.

44. ...POPULATOR - RECORDS ACCESSED OUT OF SEQUENCE.

Cause: A record was retrieved whose reference code was less than or equal to the preceding record's reference code.

Action: There are several reasons why the reference codes are not in ascending order. That situation is left to the user to correct, after which the entire Reader should be rerun.

45. ...POPULATOR - ROUTINE POPLAT ADBMS ERROR# n.

Cause: See #3.

Action: See #3.

46. ...POPULATOR - TABLES NOT INITIALIZED.

Cause: The DRT was not successfully initialized.

Action: Unless this message was preceded by other errors which can be corrected, this error can be corrected only by Data Translation personnel.

47. ...POPULATOR - TROUBLE INITIALIZING DRT.

Cause: This error message will probably be preceded by other error messages.

Action: Follow the action detailed for the other errors. If there were no preceding error messages, be sure the SDDL tables were created correctly.

48. ...POPULATOR - UNION HAS INTERNAL ERROR IN DRT.

Cause: This error message will be preceded by other error messages.

Action: See #3 and #35.

APPENDIX D

RESTRUCTURER ERROR MESSAGES

This appendix contains a list of all Restructurer error messages alphabetized by

1) numeric characters
2) alphabetic characters
3) special characters

Restructurer errors are of two types. Non-fatal errors allow the Restructurer to continue executing in order to detect as many problems as possible in one run. Fatal errors terminate the run immediately, since they are usually of a nature that makes further execution pointless (e.g., unable to open a database). Some error conditions are recoverable; for example, if the Restructurer buffer space is not sufficient to access a group of compatible access paths all at once, it will attempt to access as many together as it has room for and will access the rest separately.

An error which the Restructurer is unable to recover from causes the cancellation of Activity 03, the $UTILITY copy of the Target RIF database back to the permanent file. In the error-explanations that follow, assume this is the case unless it is mentioned specifically that the results of the run will still be consistent.

In the explanations that follow, "Cause" describes the probable cause of the error condition. "Action" describes the action taken by the Restructurer if the particular error condition occurs.

1. ERROR IN COMPAR. IPOS=_____, IERR=_____

Cause: An error has occurred while performing qualification on a source record instance.

Action: More error messages will follow.

2. ERROR IN CONSTR. IPOS=_____, IERR=_____

Cause: An error has occurred while attempting to build a target record instance.

Action: More error messages will follow.

3. ERROR IN CONVRT. IPOS=_____, IERR=_____

Cause: An error has occurred while item conversion (e.g., integer-to-character type) was being performed prior to assigning data to a target item.

Action: More error messages will follow.

4. ERROR IN INTQL1. IPOS=_____, IERR=_____

Cause: An error has occurred while performing an inter-item qualification.

Action: More error messages will follow.

5. ERROR IN INTQL2. IPOS=_____, IERR=_____

Cause: An error has occurred while performing an inter-item qualification.

Action: More error messages will follow.

6. ERROR IN QUAL. IPOS=_____, IERR=_____

Cause: An error has occurred while performing qualification on a source record instance.

Action: More messages will follow.

7. ERROR IN SITMMV. IPOS=_____, IERR=_____

Cause: An error has occurred while assigning source item values to their corresponding target items.

Action: More messages will follow.

7.5 STATISTICS INCOMPLETE DUE TO LACK OF STORAGE

Cause: Because of a lack of storage for statistics, the Restructurer was unable to compile statistics for the entire run. While the statistical summary will be incomplete, this in no way affects the success or failure of the run.

Action: Normal execution of the wrapup procedure continues.

8. ######END-OF-FILE DETECTED ON USER INPUT--#END-USER-INPUT CARD SIMULATED######

Cause: #END-USER-INPUT Directive was not used. This is merely a warning and not an error.

Action: End of User-Input Directives is assumed.

9. ******ABORT******ABORT******...ABORT******

Cause: An error has occurred during the Restructurer run that caused the operating system to terminate it (e.g., processor time exceeded), or job aborted by user. Results of the run will be inconsistent.

Action: The Restructurer abort procedure is executed and a statistical summary is printed. Activity 03 is cancelled.

10. ******ADBMS ERROR IN UINPUT; RETURN CODE=_____******

Cause: An ADBMS error occurred while processing the User-Input Directives. Consult Data Translation Project personnel.

Action: More error messages will follow.

11. ******DATABASE INITIALIZER ERROR IN UINPUT; RETURN CODE=_____******

Cause: ADBMS Database Initializer was unable to initialize the target RIF database. Consult Data Translation Project personnel.

Action: More error messages will follow.

12. ******DDL ANALYZER ERROR IN UINPUT; RETURN CODE=_____******

Cause: DDL output by ADBMS DDL Writer is inconsistent. Consult Data Translation Project personnel.

Action: More error messages will follow.

13. ******DDL WRITER ERROR IN UINPUT; RETURN CODE=_____******

Cause: Either (1) inconsistencies in target SDDL tables, or (2) internal logic error in ADBMS DDL-Writer. If (1) can be ruled out, consult Data Translation Project personnel.

Action: More error messages will follow.

14. ******ERROR IN MAIN CONTROL--ABNORMAL RETURN CODE=_____ FROM STKBLD******

Cause: The Restructurer Main Control has received an uninterpretable return code from the Stack Builder subroutine. Core may have been accidentally written. Results of the run may be inconsistent. Consult Data Translation Project personnel.

Action: The normal wrapup procedure is executed and a statistical summary is printed. Activity 03 is cancelled.

15. ******ERROR IN MAIN CONTROL--DINT UNSUCCESSFUL; RETURN CODE=_____******

Cause: The Restructurer Main Control was unable to initialize the database management system, ADBMS. This indicates a serious problem in the ADBMS routines in the Translator Library; e.g., it may not have been restored correctly from the systen tape. If this is not the case, consult Data Translation Project personnel.

Action: The Restructurer run is cancelled, along with Activity 03. No further execution takes place.

16. ******ERROR IN MAIN CONTROL--NAME TO POINTER CONVERSION ERROR; RETURN CODE=_____******

Cause: An error occurred while converting ADBMS names in the TDL Tables to their symbolic pointers; see previous messages on Restructurer Report.

Action: The Restructurer run is cancelled, along with Activity 03. No further execution takes place.

17. ******ERROR IN MAIN CONTROL--UNABLE TO CLOSE ALL DATABASES; RETURN CODE=_____******

Cause: The Restructurer Main Control was unable to close all databases previously opened. May be caused by unsuccessful earlier attempt to open a physically inconsistent database. Any results from the run are probably inconsistent.

Action: Activity 03 is cancelled to preserve the previous contents of the Target RIF database.

18. ******ERROR IN MAIN CONTROL--UNABLE TO OPEN NAMES TDL TABLES; RETURN CODE=_____******

Cause: The Restructurer Main Control was unable to open the "Names" copy of the TDL Tables. The permanent copy of the TDL Tables is probably physically inconsistent and should be re-generated.

Action: The Restructurer run is cancelled, along with Activity 03. No further execution takes place.

19. ******ERROR IN MAIN CONTROL--UNABLE TO OPEN POINTERS TDL TABLES; RETURN CODE=_____******

Cause: Same as error #18, except that it applies here to the "Pointers" copy of the TDL Tables.

Action: Same as error #18.

20. ******ERROR IN MAIN CONTROL--UNABLE TO OPEN SRIF; RETURN CODE=_____******

Cause: The Restructurer Main Control was unable to open the Source RIF database. It is probably physically inconsistent and should be re-generated.

Action: The Restructurer run is cancelled, along with Activity 03. No further execution takes place.

21. ******ERROR IN MAIN CONTROL--UNABLE TO OPEN TRIF; RETURN CODE=____******

Cause: Same as error #20, except that here it applies to the Target RIF database.

Action: Same as error #20.

22. ******ERROR IN MAIN CONTROL--USER-INPUT PROCESSING UNSUCCESSFUL; RETURN CODE=____******

Cause: An error occurred while processing the User-Input Directives; see previous messages on Restructurer Report.

Action: The Restructurer run is cancelled, along with Activity 03. No further execution takes place.

23. ******ERROR IN NTOPTR--BAD (A) IN TDL TABLES; SGNAME/SSETNM= (B) / (C) ******

Cause: An inconsistent ADBMS name from the Source RIF database has been found in the TDL Tables while attempting to convert it to its symbolic pointer. If (A) is "SGNAME", the bad name was a source record type name; if (A) is "SSETNM", the bad name was a source set type name. The (B)/(C) combination gives the source record name (B) and the source set name (C) that caused the error. The most likely cause of this error is failure to re-generate one or both of the TDL Tables and Source RIF database after the SDDL Tables have been re-generated with changes. Both the TDL Tables and the Source RIF database should be re-generated before attempting to re-run the Restructurer.

Action: Same as error #24.

24. ******ERROR IN NTOPTR--BAD (A) IN THE TDL TABLES; TGNAME/SYSSET= (B) / (C) ******

Cause: An inconsistent ADBMS name from the Target RIF database has been found in the TDL Tables while attempting to convert it to its symbolic pointer. If (A) is "TGNAME", the bad name was a target record type name; if (A) is "SYSSET", the bad name was a target system set type name. The (B)/(C) combination gives the target record name (B) and associated system set name (C) that caused the error. The most likely cause of this error is failure to re-generate the TDL Tables after the SDDL Tables have been re-generated with changes. TDL Tables should be re-generated before attempting to run the Restructurer again.

Action: Name to pointer conversion continues in order to detect all errors in one run. This messages will be followed by errors #27 and #16.

25. ******ERROR IN NTOPTR--BAD SINAME IN TDL TABLES= (A) ; SGNAME/SSETNM= (B) / (C) ******

Cause: An inconsistent ADBMS name from the Source RIF database has been found in the TDL Tables while attempting to convert it to its symbolic pointer. In this case, it is a bad source item name, given by (A). The (B)/(C) combination gives the source record type (B) of which it is an item, and the source set name (C) associated with the source record name by virtue of being a set along which (B) is accessed in a particular access path. See error #23 for likely cause.

Action: Same as error #24.

26. ******ERROR IN NTOPTR--BAD TINAME IN TDL TABLES= (A) ; TGNAME= (B) ******

Cause: An inconsistent ADBMS name from the Target RIF database has been found in the TDL Tables while attempting to convert it to its symbolic pointer. In this case, it is a bad target item name, given by (A); (B) gives the target record type of which (A) is an item. See error #24 for likely cause.

Action: Same as error #24.

27. ******ERROR IN NTOPTR--THE ABOVE ADBMS NAMES FROM THE TDL TABLES ARE INCONSISTENT******

Cause: Inconsistent ADBMS name(s) from the TDL Tables have been found in the TDL Tables while attempting to convert them to their symbolic pointers. See previous messages on Restructurer Report.

Action: Name to pointer conversion is done. This message will be followed by error #16.

28. ******ERROR IN RSTABT--UNABLE TO CLOSE ALL DATABASES; RETURN CODE= _____ ******

Cause: The Restructurer abort routine was unable to close all open databases following an error that produced error #9. Results of the run will be inconsistent.

Action: The run is terminated by the operating system.

29. ******ERROR IN SRCACC--10 NON-FATAL ERRORS HAVE OCCURRED******

Cause: 10 non-fatal errors have occurred while accessing the current stack (set of access paths).

Action: Same as error #33.

30. ******ERROR IN SRCACC--CANNOT GET FIRST INSTANCE AT LEVEL_____; RETURN CODE=_____******

Cause: The Restructurer was unable to retrieve the first source record instance on some source set. Source RIF database is probably inconsistent.

Action: The condition is treated as if there were no records on the particular source set, and the Restructurer attempts to continue accessing the current stack (set of access paths).

31. ******ERROR IN SRCACC--CANNOT GET NEXT INSTANCE AT LEVEL_____; RETURN CODE=_____******

Cause: The Restructurer was unable to retrieve the next source record instance on some source set. Source RIF database is probably inconsistent.

Action: The condition is treated as if there were no more records on the particular set, and the Restructurer attempts to continue accessing the current stack (set of access paths).

32. ******ERROR IN SRCACC--CONSTRUCTOR ERROR AT LEVEL_____; RETURN CODE=_____******

Cause: An error has occurred while attempting to construct a target record instance.

Action: The Restructurer attempts to continue accessing the current stack (set of access paths).

33. ******ERROR IN SRCACC--MULTIPLE DATABASE ERROR #_____******

Cause: An error occurred while manipulating ADBMS database currency during the accessing of an access path.

Action: Accessing of the current stack (access paths) is terminated. Restructuring continues with the next set of access path(s). This message will be followed by error #36.

34. ******ERROR IN SRCACC--NO APPTRS AT SOURCE STACK LEVEL_____******

Cause: A Restructurer internal logic error has occurred. Consult Data Translation Project personnel.

Action: Same as error #33.

35. ******ERROR IN SRCACC--QUALIFIER ERROR AT LEVEL_____; RETURN CODE=_____******

Cause: An error has occurred while attempting to perform qualification on a source record instance.

Action: The condition is treated as if the record had failed qualification, and the Restructurer attempts to continue accessing the current stack (set of access paths).

36. ******ERROR IN SRCACC--SOURCE ACCESSOR ABORTED******

Cause: Either 10 non-fatal errors or 1 fatal error have occurred while accessing the current stack (set of access paths).

Action: Processing of the current stack terminates. More error messages will follow.

37. ******ERROR IN SRCACC--UNABLE TO MOVE SRIF SYSTEM RECORD TO WORK AREA; RETURN CODE=_____******

Cause: Inconsistency exists in the Source RIF database. Consult Data Translation Project personnel.

Action: Same as error #33.

38. ******ERROR IN STAKAP--ADBMS ERROR #_____******

Cause: An ADBMS error has occurred while attempting to stack a particular access path.

Action: More error messages will follow.

39. ******ERROR IN STAKAP--IMPOSSIBLE RETURN CODE FROM STAKND=_____******

Cause: An uninterpretable error return code has been returned by a lower level subroutine while attempting to stack a particular access path. Core may have been accidentally overwritten.

Action: More error messages will follow.

40. ******ERROR IN STAKAP--UNABLE TO STACK ACCESS PATH; APPTR STORAGE EXHAUSTED*****

Cause: The Restructurer was unable to put a particular access path on the stack due to lack of internal buffer space.

Action: More error messages will follow.

41. ******ERROR IN STAKAP--UNABLE TO STACK ACCESS PATH; ROOT KEY=_____******

Cause: An error has occurred while attempting to stack a particular access path.

Action: More error messages will follow.

42. ******ERROR IN STAKAP--UNABLE TO UNSTACK PARTIAL ACCESS PATH******

Cause: An error has occurred in an error recovery procedure which was attempting to salvage the logically consistent part of a stack (set of access paths).

Action: More error messages will follow.

43. ******ERROR IN STAKND--ADBMS ERROR #_____, SWITCH=_____******

Cause: An ADBMS error has occurred while attempting to put a source record type from an access path on the stack.

Action: More error messages will follow.

44. ******ERROR IN STAKND--APPTR STORAGE EXHAUSTED******

Cause: Same as error #47.

Action: More error messages will follow.

45. ******ERROR IN STAKND--BAD APPTR INDEX RETRIEVED FROM NODE RECORD_____; INDEX=_____******

Cause: Same as error #49.

Action: More error messages will follow.

46. ******ERROR IN STAKND--NODE RECORD NOT A MEMBER OF AN INSTANCE OF EITHER OM OR MO; SWITCH=_____******

Cause: Same as error #49.

Action: More error messages will follow.

47. *******ERROR IN STAKND--SRCSTK STORAGE EXHAUSTED******

Cause: The Restructurer was unable to put a source record type from an access path on the stack due to lack of internal buffer space.

Action: More error messages will follow.

48. ******ERROR IN STAKND--UNABLE TO STACK NODE=_____******

Cause: An error has occurred while attempting to put a source record type from an access path on the stack.

Action: More error messages will follow.

49. ******ERROR IN STAKND--TDL DDL INDICATES NODE RECORD NOT LEGAL MEMBER OF OM AND/OR MO; SWITCH=_____******

Cause: Inconsistency exists in the TDL Tables. Consult Data Translation Project personnel.

Action: More error messages will follow.

50. ******ERROR IN STAPRT--ADBMS ERROR #_____******

Cause: An ADBMS error has occurred during the wrapup procedure. Results of the run may be inconsistent.

Action: Wrapup procedure is terminated at the point of the error.

51. ******ERROR IN STAPRT--MULTIPLE DATABASE ERROR #_____******

Cause: An error has occurred while manipulating ADBMS database currency during the wrapup procedure. Results of the run may be inconsistent.

Action: Wrapup procedure is terminated at the point of the error.

52. ******ERROR IN STKBLD--ADBMS ERROR #_____******

Cause: An ADBMS error has occurred during the stacking of access path.

Action: The current access path is discarded and restructuring continues with the next access path.

53. ******ERROR IN STKBLD--IMPOSSIBLE RETURN CODE FROM_____=_____******

Cause: A lower level routine has returned an uninterpretable error return code while preparing a stack. Core may have been accidentally overwritten.

Action: Same as error #55.

54. ******ERROR IN STKBLD--INCOMPLETE BUT USABLE STACK; RETURN CODE=_____******

Cause: An incomplete but logically consistent stack has been prepared. Results of the run may be inconsistent. Study Statistical Summary carefully, along with any other error messages on the report. Consult Data Translation project personnel if necessary.

Action: Restructuring continues with the current stack.

55. ******ERROR IN STKBLD--MULTIPLE DATABASE ERROR #_____******

Cause: An error has occurred while manipulating ADBMS database currency during the stacking of an access path.

Action: Restructuring stops at the point of the error.

56. ******ERROR IN STKBLD--STACK ABORTED******

Cause: Errors have occurred rendering the current stack useless.

Action: The stack is discarded. Restructuring continues with the next access path.

57. ******ERROR IN STKBLD--STACK BUILDER ABORTED******

Cause: Serious errors have occurred that warrant termination of the Restructurer run.

Action: The run is prematurely terminated.

58. ******ERROR IN STKBLD--STACK NO._____ NOT ACCESSED SUCCESSFULLY; RETURN CODE=_____******

Cause: The indicated stack was not processed successfully. Results from the corresponding access paths are probably inconsistent.

Action: Restructuring continues with the next access path.

59. ******ERROR IN STKBLD--UNABLE TO ALLOCATE WORK AREA STORAGE; RETURN CODE=_____******

Cause: The Restructurer was unable to allocate work area storage for the current stack.

Action: Same as error #52.

60. ******ERROR IN STKBLD--UNABLE TO DUMP STACK NO._____; RETURN CODE=_____******

Cause: Restructurer internal locic error. Consult Data Translation Project personnel.

Action: Same as error #55.

61. ******ERROR IN STKBLD--UNABLE TO INITIALIZE WORK AREA; RETURN CODE=_____******

Cause: The Restructurer was unable to initialize the ADBMS work area. Consult Data Translation Project personnel.

Action: Restructuring stops at the point of the error.

62. ******ERROR IN STKBLD--UNABLE TO STACK FIRST ACCESS PATH; RETURN CODE=_____******

Cause: The Restructurer was unable to put the first access path selected on the stack.

Action: Restructuring stops at the point of the error.

63. ******ERROR IN STKCMP--ADBMS ERROR #_____******

Cause: Same as error #52.

Action: Same as error #52.

64. ******ERROR IN STKCMP--IMPOSSIBLE RETURN CODE FROM STAKAP=_____******

Cause: Same as error #53.

Action: Same as error #53.

65. ******ERROR IN STKCMP--STACK HAS LOST ITS INTEGRITY; FIRST ROOT=_____******

Cause: Errors have occurred rendering the current stack useless.

Action: Same as error #52.

66. ******ERROR IN STKCMP--UNABLE TO STACK ALL COMPATIBLE ACCESS PATHS FOR FIRST ROOT=_____; STACK STILL USABLE******

Cause: Errors have occurred preventing the stacking (together) of a group of compatible access paths.

Action: Restructuring continues with a partial stack.

67. ******ERROR IN STKDMP--BAD APPTR INDEX DETECTED IN SOURCE STACK,=_____******

Cause: Same as error #60.

Action: Same as error #60.

68. ******ERROR IN STKDMP--BAD STKBTM PASSED=_____******

Cause: Same as error #60.

Action: Same as error #60.

69. ******ERROR IN STKPRP--ADBMS ERROR #_____******

Cause: Same as error #52.

Action. Same as error #52.

70. ******ERROR IN STKPRP--MULTIPLE DATABASE ERROR #_____******

Cause: Same as error #55.

Action: Same as error #55.

71. ******ERROR IN STKPRP--WORK AREA STORAGE EXHAUSTED******

Cause: Initialized work area space is insufficient for the current stack.

Action: Error recovery is initiated. See error #86.

72. ******ERROR IN UINPUT--ACTIVE-TDLAPS CARD ALREADY ENCOUNTERED; CARD WAS--

Cause: User Input Processor has read a second #ACTIVE-TDLAPS directive; only one is allowed.

Action: The bad card is printed following the message. Processing continues to find more errors, if any.

73. ******ERROR IN UINPUT--EXECUTION-MONITOR CARD ALREADY ENCOUNTERED; CARD WAS--

Cause: User Input Processor has read a second #EXECUTION-MONITOR= directive; only one is allowed.

Action: Same as error #72.

74. ******ERROR IN UINPUT--EXPECTING HASH INFO AFTER USER-HASH-INPUT CARD; CARD WAS--

Cause: #USER-HASH-INPUT directive was not followed by any hash input statements.

Action: Same as error #72.

75. ******ERROR IN UINPUT--EXPECTING HASH INFO AFTER USER-HASH-INPUT CARD; GOT EOF******

Cause: #USER-HASH-INPUT directive was last line of User-Input Directives; it must be followed by at least one hash input statement.

Action: End of User-Input Directives is assumed.

76. ******ERROR IN UINPUT--EXPECTING RESTRUCTURER CONTROL CARD; CARD WAS--

Cause: User-Input Processor was expecting a User-Input Directive, but the input line read was not a legal directive. Most probable cause is a TDLAP name following the ALL-TDLAPS-ACTIVE keyword.

Action: Same as error #72.

77. ******ERROR IN UINPUT--EXPECTING TDLAP NAME AFTER ACTIVE-TDLAPS CARD; CARD WAS--

Cause: A User-Input Directive immediately followed the #ACTIVE-TDLAPS directive. At least one TDLAP name or ALL-TDLAPS-ACTIVE must follow #ACTIVE-TDLAPS.

Action: Same as error #72.

78. ******ERROR IN UINPUT--EXPECTING TDLAP NAME AFTER ACTIVE-TDLAPS CARD; GOT EOF******

Cause: #ACTIVE-TDLAPS directive was last line of User-Input Directives. It must be followed by at least one TDLAP name or ALL-TDLAPS-ACTIVE.

Action: Same as error #75.

79. ******ERROR IN UINPUT--ILLEGAL OPTION ON EXECUTION-MONITOR CARD; CARD WAS--

Cause: EXECUTION-MONITOR=directive not followed by either ON or OFF with no intervening blanks. These are the only two options available.

Action: Same as error #72.

80. ******ERROR IN UINPUT--ILLEGAL OPTION ON RUN CARD; CARD WAS--

Cause: #RUN=directive not followed by either INITIALIZE or CONTINUE with no intervening blanks. These are the only two options available.

Action: Same as error #72.

81. ******ERROR IN UINPUT--ILLEGAL RESTRUCTURER CONTROL CARD; CARD WAS--

Cause: The input line read was not a legal User-Input Directive. Probably misspelled or contains imbedded blanks.

Action: Same as error #72.

82. ******ERROR IN UINPUT--RUN CARD HAS ALREADY BEEN ENCOUNTERED; CARD WAS--

Cause: User-Input Processor has read a second #RUN=directive; only one is allowed.

Action: Same as error #72.

83. ******ERROR IN UINPUT--TDLAP NAME (A) NOT FOUND ON ROOTS SET******

Cause: The user has requested TDLAP (A) to be active; however, no such TDLAP name exists in the TDL tables. Check TDL description for correct spelling.

Action: Processing continues to find more errors, if any.

84. ******ERROR IN UINPUT--USER-HASH-INPUT CARD ALREADY ENCOUNTERED; CARD WAS--

Cause: User-Input Processor has read a second #USER-HASH-INPUT directive; only one is allowed.

Action: Same as error #72.

85. ******ERROR RECOVERY FAILED******

Cause: The error condition associated with errors #71 and #86 could not be recovered from. This may result from too many access paths being compatible. Restructuring-by-parts should be used; i.e., only a few active access paths per run.

Action: More error messages will follow.

86. ******ERROR RECOVERY INITIATED; WORK AREA RE-INITIALIZED WITH_____ PAGES******

Cause: Lack of initialized work area space.

Action: More work area space is initialized. Either error #87 or #85 should follow this message.

87. ******ERROR RECOVERY SUCCESSFUL******

Cause: The error condition associated with errors #71 and #86 has been successfully recovered from.

Action: Restructuring continues. Results of the run will be good as far as this error condition is concerned; i.e., Activity 03 will not be cancelled.

88. ******ERROR(S) IN PHASE II --ABNORMAL TERMINATION******

Cause: Fatal error(s) during Phase II of the Restructuring run caused premature termination of execution. Results of the run are probably inconsistent.

Action: The normal wrapup procedure is executed and a Statistical Summary is printed. Activity 03 is cancelled.

89. ******ERROR(S) IN PHASE II, BUT ABLE TO TERMINATE NORMALLY******

Cause: Non-fatal error(s) occurred during Phase II of the Restructurer run, but none serious enough to warrant premature termination of the run. Results of the run are probably inconsistent.

Action: The normal wrapup procedure is executed and a Statistical Summary is printed. Activity 03 is cancelled.

90. ******EXECUTION DISASTER IN NAME TO POINTER CONVERSION LINE#=_____, IERR=_____ .

Cause: An error has occurred in an ADBMS subroutine while converting ADBMS names in the TDL Tables to their symbolic pointers. TDL Tables may be physically inconsistent and should be re-generated. If this is not the case, consult Data Translation Project personnel.

Action: Name to pointer conversion terminates immediately. This message will be followed by error #16.

91. ******MULTIPLE DATABASE ERROR IN UINPUT; RETURN CODE=_____******

Cause: An error occurred while manipulating ADBMS database currency during processing of the User-Input Directives. Consult Data Translation Project personnel.

Action: Processing of the User-Input Directives terminates immediately. More error messages will follow.

92. ******RESTRUCTURER ABORT PROCEDURE DONE******

Cause: This message indicates that the Restructurer abort procedure finished (see error #9).

Action: The run is terminated by the operating system.

93. ******RESTRUCTURER RUN CANCELLED******

Cause: A fatal error has occurred in Phase I of the Restructurer run; see previous messages on Restructurer Report.

Action: Self-explanatory; no restructuring will be performed.

94. ******WARNING--TRANS-NAME CARD ALREADY ENCOUNTERED; FOLLOWING CARD IGNORED--

Cause: User-Input Processor has read a second #TRANSLATION-NAME=directive; only the first one is accepted. This is not an error, only a warning.

Action: Same as error #72.

APPENDIX E

WRITER ERRORS

Errors in the Writer can be divided into three classes according to their degree. The most serious cause the Writer immediately to terminate with a FORTRAN STOP statement. Less serious errors will print a message, cause an error recovery action to be initiated which, in turn, continues execution. These errors generally have a multiplicative effect and force more errors as the Writer continues to execute. Finally, there are warnings which generally refer to problems with items within records. For almost every application, all nonfatal errors and warnings must be corrected, thus, forcing a rerun of the Writer and, in some extreme cases, a rerun of other Translator modules. Sophisticated users can correct some minor problems by writing a special-purpose program to alter the target database if a rerun of the Writer cannot be justified.

Documentation for all Writer errors appears in this appendix. Each error message is given a number and cross-references are sometimes made. Information as to whether or not the error is fatal/nonfatal, which activity the error occurs in, cause, the action to be taken to correct each error, are given. All occur in activity #4 and are sorted alphabetically to allow easy reference. The collating sequence followed is:

1) Numeric characters.
2) Alphabetic characters.
3) Special characters.

1. "RECORD TYPE NOT STORED IS - _____ PROBABLE CAUSE - ALL OF ITS MASTER RECORDS HAVE NOT BEEN STORED YET."

Cause: A run of the Writer on only part of the database has failed to store the record type mentioned in the error message because the master of this record type was not specified by the user to be written on this or a prior run. This is a nonfatal error.

Action: Rerun the Writer by including the master of this record type among those to be written, or remove this record type from the list of those to be written.

2. "###ERROR...ADBMS ERROR OCCURRED DURING RETRIEVER ATTEMPT TO GET RECORD-____FIELD NAME-____WITH DBKEY-____"

Cause: Accompanied by the ADBMS trace listing on RC-06, a Writer initializing routine suffered an ADBMS error. This fatal error is a sign of seriously damaged target SDDL tables and/or a program logic error.

Action: Make sure the IDS Analyzer run that produced the target SDDL tables terminated normally with no errors. It is doubtful that the Restructurer or TDL Analyzer could have executed successfully if this error occurs. If all else fails, contact the Data Translation Project.

3. "###ERROR...ADBMS ERROR OCCURRED WHILE ATTEMPTING TO LOCATE DETAIL TYPE OF THESE PREVIOUSLY STORED MASTERS - IDS REC TYPE STORED FLAG"

Cause: The Writer was attempting to determine which detail type was to be stored next when this fatal error occurred. The following is a list of possible causes:

1. The SDDL tables have been damaged as in error #2.
2. The user has changed the record type identification numbers between compiling the target MD section in activity 3, and the time that the target SDDL tables were created by the IDS Analyzer.
3. A disjoint, invalid IDS structure was defined in the target MD section.

Action: The first two causes are self-explanatory; the third requires examination of the error message. Draw a picture of the database marking the records with an "X" if stored by this point (indicated by a "1" in the STORED column). There must be at least one record that is a detail of only records marked with an "X", if not, an invalid IDS structure has been defined.

4. "###ERROR...ADBMS ERROR OCCURRED WHILE ATTEMPTING TO LOCATE OWNER RECORDS (GETOWN) FOR TYPE - ____"

Cause: See error #2.

Action: See error #2.

5. "###ERROR...DATABASE NAME ____ NOT FOUND###"

Cause: The name of the database supplied to the Writer by the user as a run-time parameter is invalid. This is a fatal error.

Action: Check to be sure that the database name is spelled correctly and that this database is described by the SDDL tables included in the JCL.

6. "###ERROR...DBNAME IS EMPTY###"

Cause: The user has not supplied the name of the database the Writer is to work on as a run-time parameter. This is a fatal error.

Action: Correct the JCL so that the first line of the data file assigned to I/O device 15 is the name of a database consistent with its name in the SDDL tables.

7. "###ERROR...ERROR IN GTSYOW, CAN'T FIND SYSTEM SET NAME FOR-____"

Cause: Certain records in the SDDL tables were built incorrectly or incompletely. Alternatively, the Writer has damaged itself during execution. This is a nonfatal error.

Action: Check the IDS Analyzer for run errors. If none can be found, contact Data Translation personnel.

8. "###ERROR...ERROR IN MODCHT, UNABLE TO MODIFY CHAIN TABLES FOR MASTER RECORD IDENTIFIED BY-____ FOR CHAIN-____"

Cause: An invalid IDS Definition Structure was created because an erroneous MD section was written. This is a nonfatal error.

Action: Ensure that the compilation of the MD section in activity 3 was correct. See also solution for error #9.

9. "###ERROR...FDMOD DISCOVERED INVALID IDS STRUCTURE TABLE, ERROR# = ____"

Cause: An invalid IDS structure was defined in the target MD. The IDS Translator does not catch all semantic errors in the coding. The error # is:

1 - No RDs in structure table
3 - A field increment and field length>3813
4 - Last FD pointer does not point to an RD
5 - Last RD pointer does not point to CCBLC

It is possible that if activity 3 was not checked for errors this message will appear. It is a fatal error.

Action: Check activity 3 for errors. Check also to make certain that each 98 CHAIN MASTER entry has at least one 98 CHAIN DETAIL entry. Carefully examine the MD section for syntax errors that IDS might not detect.

10. "###ERROR...GETNXT ROUTINE RETURNED ADBMS ERROR OF-____, ATTEMPTED TO RETRIEVE TYPE-____"

Cause: See error #2. This is a nonfatal error.

Action: See error #2.

11. "###ERROR...GETRD SUFFERED SOME ERROR, FOR IDSREC-____, RECORD NAME = ____"

Cause: A Writer initializing routine could not locate a record definition for the given record name. It was probably caused by an invalid IDS Definition Structure. This is a fatal error.

Action: Same as error #9.

12. "###ERROR...GETREF HAD ADBMS ERROR OCCUR FOR DBKEY-____ WHILE LOOKING FOR OWNER OF DETAIL TYPE-____. OWNER = ____, SET = ____. DETAIL INSTANCE IGNORED"

Cause: The record was not linked up on the set specified by the Restructurer. This error almost always occurs because the TDL was not written correctly for the application. Coupled with error #38. This is a nonfatal error.

Action: Check the TDL, rewrite and rerun the TDL Analyzer and Restructurer.

13. "###ERROR...GETTOP HAD A ADBMS ERROR INDICATED BY CODE #-____."

Cause: See error #2. This is a fatal error.

Action: See error #2.

14. "###ERROR...IMPOSSIBLE TO INITIALIZE ADBMS DATA BASES. ERROR CODE = ____, EXECUTION TERMINATED"

Cause: Improper JCL for executing the Writer. This is a fatal error.

Action: Check file code assignments as specified in Section 9.7.

15. "###ERROR...IMPOSSIBLE TO LOCATE MASTER DEF'N FOR CHAIN-____"

Cause: Ths user has substituted or changed the target MD section between the creation of the target SDDL tables and the execution of the Writer. In particular, the location of the chain pointer fields has not remained constant. This is a nonfatal error.

Action: Make sure the 98 level entries have not been transposed since the IDS Analyzer run. All item lengths should remain constant. If an error is found, rerun the Writer after correcting the MD.

16. "###ERROR...IMPOSSIBLE TO OPEN TARGET SDDL"

Cause: Improper JCL for executing the Writer. This is a fatal error.

Action: Check file code assignments specified in Section 9.7.

17. "###ERROR...IMPOSSIBLE TO STORE REFERENCE CODE. IERR RETURNED WITH A POSITIVE VALUE FROM SFK. IERR = ____"

Cause: Creation of the target RIF did not execute successfully. Error in the DDL Writer during Restructurer run. This is a nonfatal error.

Action: Check the output of the DDL Writer from the Restructurer for errors. Erroneous information means that the target SDDL tables were not properly created. Refer to the section on the IDS Analyzer.

18. "###ERROR...IMPOSSIBLE TO STORE REFERENCE CODE OF TARGET RECORD-____ WITH DBKEY IN RIF OF-____ PROCESSING CONTINUES BUT ERRORS MAY OCCUR LATER WHEN ATTEMPTING TO STORE DETAIL RECORDS OF THIS MASTER"

Cause: See error #17. This is a nonfatal error.

Action: See error #17.

19. "###ERROR...INCONSISTENCY BETWEEN TARS ROUTINE GTDETL WHICH CLAIMED THAT TYPE-____ HAD MASTER TYPES AND ROUTINE GETOWN WHICH COULDN'T FIND ANY MASTERS.

Cause: See error #2. This is a fatal error.

Action: See error #2.

20. "###ERROR...INCONSISTENCY IN ROUTINE GETOWN, RECORD TYPE-____ WAS NOT RETURNED IN INITIALIZATION ROUTINE GETREC"

Cause: See error #2. This is a fatal error.

Action: See error #2.

21. "###ERROR...INVALID RUN-TIME INPUT, FORMAT MUST BE A6, 1X, A6. VALUES READ WERE - ____"

Cause: The second line of data residing on file code 15 is not in the proper format to be used as a run-time input parameter(s). This is a fatal error.

Action: Make sure that the input parameters to the Writer are as specified in Section 9.7.

22. "###ERROR...INVALID USER SPECIFIED RECORD TYPE NAME, EXECUTION CONTINUES. INVALID NAME IS ____"

Cause: The user has supplied a record type name on file code 15 which is not a record type in the target database to be created by the Writer. This is a nonfatal error.

Action: Check the Writer JCL to be sure that the record type name is spelled and formatted properly. Check also to be sure that the target database file, the SDDL tables file, and the database on file code 15 are consistent.

23. "###ERROR...LENGTH OF-____ DISP OF-_____FOR ITEM-____ OF RECORD-_____ EXCEEDS BUFFER MAX OF 3840 CHARACTERS"

Cause: While building a target record, one of the Writer's internal vectors overflowed because the sum of the record's item lengths was too large. This is a nonfatal error.

Action: Either redesign the target database so that the record will not be greater than 3840 characters or contact the Data Translation Project to modify the internal table.

24. "###ERROR...MASTERLESS RECORD TYPE OF-____ DOES NOT EXIST WITHIN DBTGNM VECTOR RETURNED BY GTRECS. EXECUTION CONTINUES"

Cause: The target SDDL tables have been seriously damaged since their creation. This error would appear only under very unusual circumstances. It is a nonfatal error.

Action: See error #2.

25. "###ERROR...NEGATIVE NUMBER ENCOUNTERED IN 'STORED' ARRAY. NUMBER IS: _____"

Cause: An internal Writer vector has an element whose value is negative. This is a fatal error.

Action: This error can occur only if the integrity of the data stored on file code 16 has been violated, or if there is a program logic error of the Writer. In either case, contact the Data Translation Project.

26. "###ERROR...NO MAPPAR ELEMENTS FOR RELAY-_____"

Cause: Error in construction of SDDL tables for target IDS database. No indication was made of the location of pointer items that implement this relation. Check SDDL table creation run and examine user dump closely. This is a nonfatal error.

Action: If no error can be found, contact Data Translation Project, otherwise correct extended target MD section, rerun the IDS Analyzer and then rerun the Writer. If the user error would have had an impact on the integrity of the Restructuring execution, then the TDL Analyzer and Restructurer must also be rerun prior to Writer execution.

27. "###ERROR...NOT POSSIBLE TO OPEN TARGET RIF"

Cause: Incorrect JCL setup for Writer execution. This is a fatal error.

Action: Check the JCL to ensure that file 02 is a random file which represents the target RIF database.

28. "###ERROR...NUMBER OF MASTER-LESS RECORD TYPES IS NOT BETWEEN 1 AND 50."

Cause: The user has violated a Writer internal limitation. There cannot be more than 50 or less than 1 record types that do not have masters, e.g., are not described by a 98 CHAIN DETAIL (other than CALC). This is a fatal error.

Action: Redesign target database so that the above restriction is observed.

29. "###ERROR...NUMBER OF TARGET IDS RECORDS IS NOT BETWEEN 1 AND 75, NUMREC = _____, IF NUMREC >75, VECTORS MUST BE REDIMENSIONED AND PROGRAM RECOMPILED"

Cause: The user has attempted to create a target database of greater than 75 record types and has thereby exceeded the Writer's internal table size. This is a fatal error.

Action: Several vectors in the Writer and the program code which checks table size overflow must be changed and the program recompiled. This should be done only by Data Translation personnel. An easier solution is to design a smaller target database.

30. "###ERROR...NUMBER OF UNIQUE FIELDS IN RECORD TYPE-_____EXCEEDS WRITER MAXIMUM OF 60."

Cause: The user has violated a Writer limitation of 60 fields per record size. This is a fatal error.

Action: Redesign the target database and start over. Alternatively, contact the Data Translation Project to modify the Writer.

31. "###ERROR...OWNER OF SET-_____WAS NOT STORED, DETAIL IGNORED"

Cause: While attempting to store a record type, the Writer discovered that not all of this record type's masters had been previously stored. Therefore, the detail could not be stored. Generally a result of previous error #12's occurrence higher in the database structure. This is a nonfatal error.

Action: Make sure that the Writer is supplied with the name of the owner record of the set mentioned on the next run of the Writer.

32. "###ERROR...RECORD-_____ IS DETAIL IN GREATER THAN 10 CHAINS, EXCEEDS WRITER MAX."

Cause: A record is defined in the target database as a detail in more than 10 chain relations. This has caused a Writer table to overflow. This is a nonfatal error.

Action: Redesign the database or contact the Data Translation Project.

33. "###ERROR...ROUTINE GETTOP CLAIMS _____ IS A DETAIL RECORD BUT ROUTINE GTDETL CLAIMS THAT IT IS NOT A MEMBER OF ANY SET."

Cause: Internal Writer logic error. This is a fatal error.

Action: Contact Data Translation Project.

34. "###ERROR...ROUTINE GTDETL, _____ IS AN OWNER OF _____ BUT IS NOT IN ORIGINAL LIST OF LEGAL GROUPS"

Cause: Internal Writer logic error. This is a fatal error.

Action: Contact Data Translation Project.

35. "###ERROR...ROUTINE GTFLD SUFFERED ADBMS ERROR _____ WHILE LOOKING FOR FIELDS OF RECORD TYPE- "

Cause: Accompanied by an ADBMS traceback, this error indicates serious problems with the target SDDL tables. This is a nonfatal error.

Action: See error #2.

36. "###ERROR...SUBROUTINE VALREC HAS RETURNED AN ILLEGAL RETURN CODE. RETURN CODE = "

Cause: A parameter passed back to the Writer from a support module is illegal. This is a fatal error.

Action: This error can occur only as a result of a serious program logic error. Contact the Data Translation Project.

37. "###ERROR...SUMSCM RETURNS ERROR CODE OF- EXPLANATION, SEE SUMSCM DOCUMENTATION"

Cause: A Writer routine that modifies bits in detail definition records of the IDS Definition Structure discovered an invalid structure. This is a fatal error.

Action: Check activity 3 output for correct compiling by IDS Translator. The error is in 98 CHAIN MASTER entries. Correct and rerun the Writer.

38. "###ERROR...THE RECORD - _____ IDENTIFIED BY KEY - _____ IS NOT AN INSTANCE OF SET - _____"

Cause: See error #12. This is a nonfatal error.

Action: See error #12.

39. "###ERROR...USER RECORD TYPE HAS ALREADY BEEN WRITTEN. RECORD NAME IS: _____ DATE WRITTEN: _____"

Cause: The user has included on file code 15 the name of a record to be written into the target database on this run of the Writer, but this record type has already been processed. This is a nonfatal error.

Action: None is necessary unless a clean output listing is desired. If so, remove the offending record name from the file residing on file code 15.

40. "###ERROR...USER SUPPLIED DATABASE NAME IS NOT THE SAME AS THE ONE STORED ON FILE CODE 16. USER'S CURRENT DATABASE ON FILECODE 16 IS _____"

Cause: While attempting to run the Writer using its partial database write capability, the user has supplied the name of a database on file code 15 which does not match the name of the database residing on file code 16. This is a fatal error.

Action: Check the JCL setup to be sure that all file names supplied are consistent with Writer JCL rules and the user's intent. Make sure that the database name supplied on file code 16 is spelled correctly and is left-justified.

41. "###ERROR...WHILE ATTEMPTING TO STORE THE RECORD - _____AN IDS ERROR OCCURRED OF #-
PROCESSING CONTINUES BUT ERRORS MAY PROPAGATE WHILE CONTINUING"

Cause: Refer to the IDS manual for an explanation of the error number. Typical errors are duplicate CALC records or space overflow. This is a nonfatal error.

Action: Depending on the error, either the TDL may have to be rewritten forcing a new Restructurer run, or the target MD section must be modified by expanding page ranges or percent fill. If the latter action is taken, the Writer must be rerun.

42. "###WARNING...IDS DISPLACEMENT OF -_____ IS NOT VALID FOR ITEM NAME - _____ OF RECORD NAME - _____ FIELD IGNORED"

Cause: The displacement of an item is less than zero. Result of serious error in the creation of target SDDL tables. This is a nonfatal error.

Action: Check IDS Analyzer run for errors. If none are found, contact the Data Translation Project.

43. "###WARNING...IDS LENGTH OF-_____ IS NOT VALID FOR ITEM NAME-_____ OF RECORD NAME-FIELD IGNORED."

Cause: The length of an IDS item is less than or equal to zero. This is a nonfatal error.

Action: Check IDS Analyzer run for errors.

44. "###WARNING...ILLEGAL TYPE VALUE OF-_____ FOR ITEM-_____ OF RECORD-_____"

Cause: Generation of item type values by the IDS Analyzer was incorrect. The only legitimate values are integers 1-7. This is a nonfatal error.

Action: Check IDS Analyzer run to ensure that for all items created, a legitimate type was created.

Final Comments on Error Messages

All errors that occur while calling the ADBMS routines appear with a traceback of calls and arguments on report code 06.

Several errors in the Writer are accompanied by an octal-BCD dump of the first five or seven words of the record involved in the error. If seven words are printed, it is the IDS record and the items are stored in exactly the same sequence as they appear in the target MD section. When five words appear, they represent the first five words of a target RIF data record in which the items are stored, ordered in a non-deterministic manner.

Many of the Writer errors point to serious problems in the execution of the Translator code. It is to be expected that system programmer intervention may be required to correct the error. If the target SDDL tables have not been altered since their creation by the IDS Analyzer, many of the possible Writer errors will already have been uncovered by the TDL Analyzer or Restructurer runs.

F.0 ADBMS OVERVIEW

ADBMS, like IDS, is a database management system which facilitates the creation, maintenance and accessing of simple and complex data structures. It consists of a collection of FORTRAN-callable subroutines whose purpose is to create databases from a user's data structure description, and to serve as an interface between the user and these databases.

ADBMS is used by the Data Translator as follows:

IDS Analyzer
- a) creates SDDL tables (ADBMS database)
- b) uses Internal Work Database (ADBMS database) internally

TDL Analyzer
- a) retrieves information from source and target SDDL tables (ADBMS databases)
- b) creates TDL tables (ADBMS database)

Reader
- a) uses source SDDL tables (ADBMS database) as input
- b) creates source RIF (ADBMS database)
- c) uses DRT (ADBMS database)

Restructurer
- a) retrieves data from source RIF (ADBMS database)
- b) stores data in target RIF (ADBMS database)
- c) accesses TDL tables (ADBMS database)

Writer
- a) uses target SDDL tables (ADBMS database) as input
- b) uses target RIF (ADBMS hash database) as input.

F.1 Describing an ADBMS Database -- The DDL

Every ADBMS database is logically described in terms of sets, records, and items. These ADBMS constructs are similar to the IDS chain, record, and field constructs. Records are composed of items and are related by sets.

There are two types of ADBMS records, hash and non-hash, whose principle difference lies in the method used to store them. While non-hash records are simply stored in the next available space in the database, the storage algorithm used for hash records is similar to that of the IDS calculated record. That is, one or more items in the hash record are specified as primary key items and are hashed, or randomized, to determine the location of the record in the database.

There are also two types of ADBMS sets, ordered and match-key, which are defined in terms of their owner and member record types, just as IDS chains have master and detail record types. Ordered sets can be used to specify relationships among records and may be ordered in a variety of ways. An ADBMS ordered set differs from an IDS chain in that it is allowed more than one owner record type. The second type of ADBMS set, the match-key set, can have only one owner record type which must be a hash record. The member-to-owner relationship is established by storing the primary key items of the owner of the set in the set-significant items of the member.

The Data Description Language, or DDL, is a language used to describe a database in terms of the constructs specified above. A specific database description written in this language is also called a DDL and corresponds to the IDS MD Section.

F.2 The DDL Analyzer/Database Initializer

The DDL Analyzer/Database Initializer is a utility routine used in creating ADBMS databases. The first step in the creation of an ADBMS database is the writing of a DDL to describe it. This DDL and some optional hash input information are then input to the DDL Analyzer/Database Initializer whose purpose is twofold. First, it analyzes the DDL description of the database, and checks for syntactical errors and inconsistencies. This first stage is similar in function to that of the IDS Translator. If it is successful, the database tables are produced and used, along with the optional hash input, to initialize the ADBMS database. The second stage, comparable to the IDS program QUTI, involves the determination of the parameters to be used by the hashing algorithm to store hash records. The user has the option of specifying some or all of these parameters as hash input to the DDL Analyzer/Database Initializer, or simply accepting default values determined during initialization. Figure F-1 summarizes the function of the DDL Analyzer/Database Initializer. Alternate modes of operation allow the user to forego the database initialization stage and receive as output only the database tables, or to input the database tables instead of the DDL and bypass the DDL analysis stage.

F.3 The ADBMS Database and Database Tables

An initialized ADBMS database consists of physical pages on which all information in the database is stored. The database tables produced during the DDL analysis stage of the DDL Analyzer/Database Initializer are stored on the first page(s) of the database. They contain the logical description of the database according to the DDL and are comparable to the IDS Definition Structure. The remaining pages are initialized according to a specific format very similar to the pages of an IDS database. Before operations can be performed on a database, the database must be opened, at which point its tables are read from the database back into core where they are used to retrieve descriptive information about the database and to store additional information required by ADBMS when performing operations on the database.

ADBMS databases which contain hash records are often referred to as hash databases; those not containing hash records are referred to as non-hash databases.

F.4 Multiple Databases

ADBMS has the capability to manage multiple databases simultaneously. That is, operations can be performed first on one database and then another without closing the first database and opening the second database between operations. This capability is made possible by the concept of a current database. Rather than close one database and open another, the user keeps all necessary databases open at the same time and simply resets the current database. All database tables corresponding to open databases are in core simultaneously; one set of them is associated with the current database.

F.5 Database Keys

Each physical record stored in the database is uniquely identified by its database key. A database key specifies the page and displacement within that page where the record is located. Whenever a record is accessed, the page on which it is stored must be brought into core.

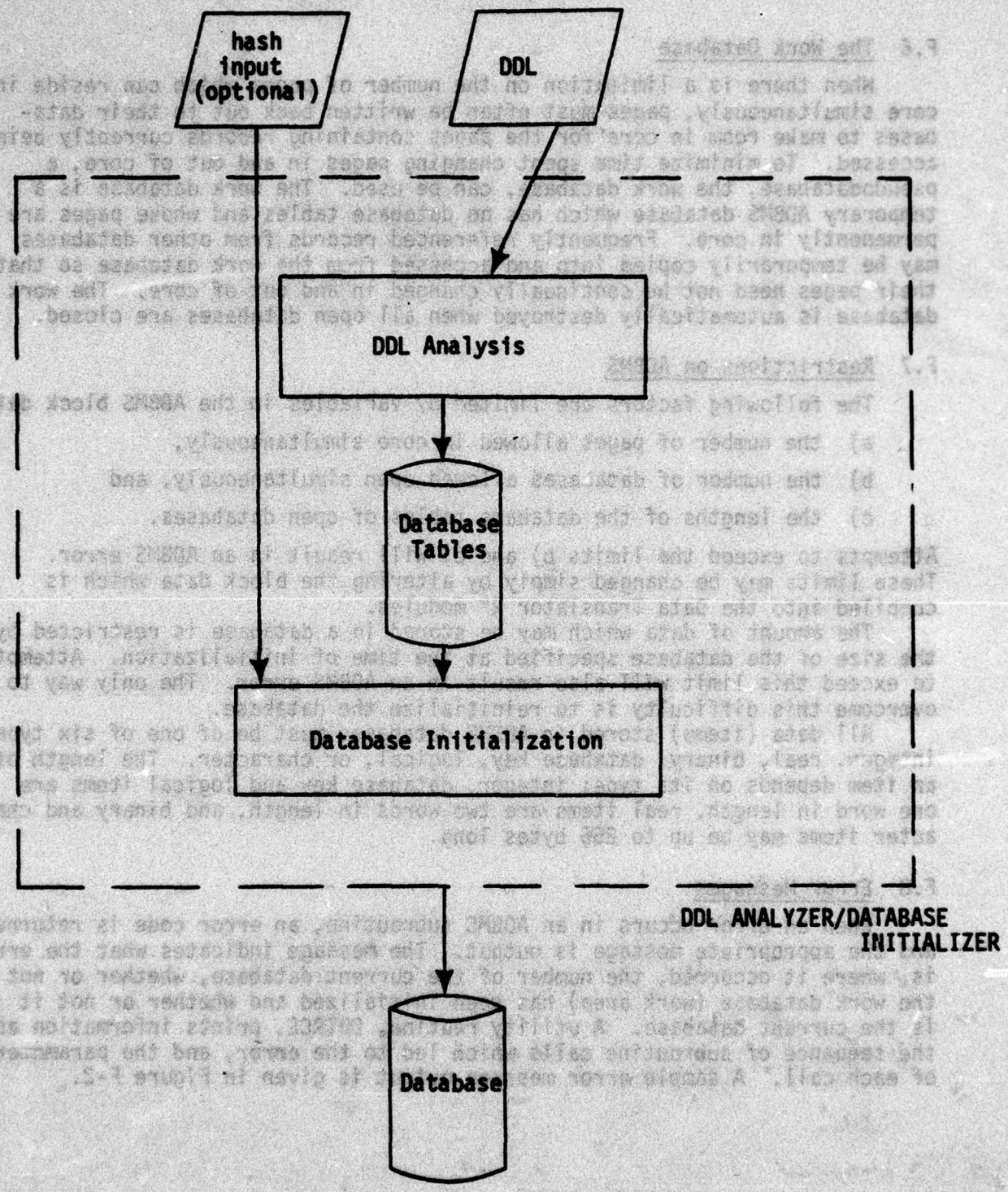


Figure F-1
DDL Analyzer/Database Initializer

F.6 The Work Database

When there is a limitation on the number of pages which can reside in core simultaneously, pages must often be written back out to their databases to make room in core for the pages containing records currently being accessed. To minimize time spent changing pages in and out of core, a pseudodatabase, the work database, can be used. The work database is a temporary ADBMS database which has no database tables and whose pages are permanently in core. Frequently referenced records from other databases may be temporarily copied into and accessed from the work database so that their pages need not be continually changed in and out of core. The work database is automatically destroyed when all open databases are closed.

F.7 Restrictions on ADBMS

The following factors are limited by variables in the ADBMS block data:

a) the number of pages allowed in core simultaneously,

b) the number of databases allowed open simultaneously, and

c) the lengths of the database tables of open databases.

Attempts to exceed the limits b) and c) will result in an ADBMS error. These limits may be changed simply by altering the block data which is compiled into the Data Translator R* modules.

The amount of data which may be stored in a database is restricted by the size of the database specified at the time of initialization. Attempts to exceed this limit will also result in an ADBMS error. The only way to overcome this difficulty is to reinitialize the database.

All data (items) stored in ADBMS databases must be of one of six types: integer, real, binary, database key, logical, or character. The length of an item depends on its type; integer, database key and logical items are one word in length, real items are two words in length, and binary and character items may be up to 256 bytes long.

F.8 Error Messages

When an error occurs in an ADBMS subroutine, an error code is returned and the appropriate message is output. The message indicates what the error is, where it occurred, the number of the current database, whether or not the work database (work area) has been initialized and whether or not it is the current database. A utility routine, DBTRCE, prints information about the sequence of subroutine calls which led to the error, and the parameters of each call. A sample error message output is given in Figure F-2.

```
ERROR #  17 IN CRX    (LEVEL   3)  DR =    4
WORK AREA INITIALIZED.  #1529
WORK AREA CURRENT.      #1530
ROUTINE  DBTRCE. STATEMENT    11    #1531
PARAMETER     ADDR          BCD       OCT            INTEGER
    1       000000070111 000000 000000000000              0
ROUTINE  DBERR . STATEMENT    99    #1534
PARAMETER     ADDR          BCD       OCT            INTEGER
    1       000000065701 000006 000000000006              6
    2       000000065702 000000 000000000000              0
ROUTINE  CRX    . STATEMENT   158    #1538
PARAMETER     ADDR          BCD       OCT            INTEGER
    1       000000130502 00001E 000000000125             85
    2       000000130503 00000A 000000000021             17
ROUTINE  STKERR. STATEMENT   118    #1542
PARAMETER     ADDR          BCD       OCT            INTEGER
    1       000000114717 0000[R 000000001251            681
    2       000000114720 000000 000000000000              0
    3       000000114721 00000A 000000000021             17
ROUTINE  STKBLD. STATEMENT   160    #1547
PARAMETER     ADDR          BCD       OCT            INTEGER
    1       000000110226 000000 000000000000              0
ROUTINE  ....... STATEMENT   153    #1550
PARAMETER     ADDR          BCD       OCT            INTEGER
    1       000000061014 000000 000000000000              0
```

Figure F-2
ADBMS Error Message Output

The following is an explanation of all possible error codes.

01 DINT HAS NOT BEEN CALLED
02 INVALID SET TYPE
03 INVALID RECORD TYPE
04 INVALID ITEM TYPE
05 INVALID OWNER TYPE
06 INVALID MEMBER TYPE
07 INVALID DATABASE KEY
08 NO CURRENT OWNER OF SET
09 NO CURRENT MEMBER OF SET
10 NO CURRENT OF RECORD TYPE
11 ALREADY MEMBER OF SET
12 RECORD NOT MEMBER OF SET
13 DEPENDED ON ITEM OUT OF RANGE
14 DATABASE ALREADY OPEN
15 DDL TABLES INCONSISTENT

16 DATABASE INCONSISTENT
17 DATABASE OVERFLOW
18 SET TYPE NOT SORTED
19 DINT HAS ALREADY BEEN CALLED
20 INVALID NUMBER OF BUFFERS
21 NO CURRENT DATABASE
22 TOO MANY DATABASES
23 ILLEGAL I/O UNIT FOR DATABASE
24 ILLEGAL I/O UNIT FOR DATABASE TABLES
25 ILLEGAL USAGE VALUE
26 DATABASE TABLE OVERFLOW
27 ILLEGAL CURRENT DATABASE
28 INVALID OPERATION ON SYSTEM RECORD
29 DATABASE OPENED FOR READ ONLY
34 INVALID ITEM POINTER
35 INVALID RECORD POINTER
36 INVALID SET POINTER
37 INVALID PAGE NUMBER
38 INVALID HASH INPUT
39 INVALID RECORD STORAGE METHOD
40 INVALID OPERATION ON PRIMARY KEY

F.9 References

The following references provide a detailed explanation of the database management system on which ADBMS is based.

E. A. Hershey III and P. W. Messink "A Database Management System for PSA Based on DBTG 71" ISDOS Working Paper no. 88, July 1975.

M. J. Bastarache and E. A. Hershey III "The Database Management Sytem User Manual and Example" ISDOS Working Paper no. 89, April 1975.

E. A. Hershey III, R. L. Dissen and P. W. Messink "A Description of ADBMS Version D210" ISDOS Working Paper no. 122, July 1975.

APPENDIX G

SYSTEM GENERATION

The Version IIA Release 2 Data Translator is supplied to the user on a Honeywell system - standard format 9-track tape. This section describes the files on the System Generation tape and the files a user must create to run the Data Translator.

The first file on the System Generation tape is the table of contents. Figure G-1 is the preliminary table of contents for the tape. After the table of contents, the tape contains the files for the IDS Analyzer, TDL Analyzer, Reader, Restructurer, and Writer. The last two files on the tape are the translation library and the DDL Writer's work database.

To bring the Data Translator up on a system the user should complete the following steps:

1. Read the entire User Manual. The entire Data Translation process should be thoroughly understood before any modules are run. All questions should be sent to the Data Translation Project.

2. Copy the first file from the tape into a 10 llink, sequential, permanent file. Convert the file to ASCII using BCDASC and print it out. The first file has the table of contents for the tape and some sample JCL for bringing the remaining files off the tape.

3. Using either FILSYS in batch or ACCESS in time-sharing, create all of the files for the Translator. Figure G-1 shows the type, length and suggested name for the files on the tape. At this point, the user should also create all files needed by each module (see Sections 5, 6, 7, 8, 9).

   Due to the large number of files used by the Data Translator, the user will find it beneficial to segregate the files used by each module. The following catalogs, created under the user's master catalog, are recommended.

Catalog	Contents
/IDSAN	IDS Analyzer files
/TDLAN	TDL Analyzer files
/RDR	Reader files
/REST	Restructurer files
/WTR	Writer files
/UTIL	Translation library, DDLWT Work database

The following suffixes should also be used on file names:

Suffix	File content
.AF	ADBMS database
.AT	ADBMS DBTF
.D	Data
.ID	IDS database
.IF	ISP index

System Generation Tape			User Permanent File		
File Number	Module	Contents	Suggested Name	Size (1links)	Mode
1	---	Table of Contents, Sample JCL	INFO	20	Seq.
2	IDS Anal.	Prototype JCL	/IDSAN/PROTO.J	2	Seq.
3	IDS Anal.	K* Source Library	---	---	Seq.
4	IDS Anal.	Phase 1 R* File	/IDSAN/PHAS1.RS	85	Seq.
5	IDS Anal.	Phase 2 R* File	/IDSAN/PHAS2.RS	12	Seq.
6	IDS Anal.	SDDL Table's DBTF	/IDSAN/SDDL.AT	2	Seq.
7	IDS Anal.	Internal DBTF	/IDSAN/INTRN.AT	2	Seq.
8	TDL Anal.	Prototype JCL	/TDL/PROTO.J	2	Seq.
9	TDL Anal.	K* Source Library	---	---	Seq.
10	TDL Anal.	Main TDLAN R* File	/TDL/TDLAN.RS	65	Seq.
11	TDL Anal.	SYSACC R* File	/TDL/SYSAC.RS	7	Seq.
12	TDL Anal.	Compatibility R* File	/TDL/CMPAT.RS	30	Seq.
13	TDL Anal.	TDL Dump R* File	/TDL/TDDMP.RS	25	Seq.
14	TDL Anal.	TDL Table's DBTF	/TDL/TDL.AT	3	Seq.
15	TDL Anal.	TDL Parsing tables	/TDL/PARSE.D	15	Seq.

Figure G-1

System Generation Tape			User Permanent File		
File Number	Module	Contents	Suggested Name	Size (11inks)	Mode
16	Reader	Prototype IDS JCL	/RDR/IDS.J	2	Seq.
17	Reader	Prototype ISP JCL	/RDR/ISP.J	2	Seq.
18	Reader	Prototype Seq. JCL	/RDR/SEQ.J	2	Seq.
19	Reader	IDS K* Source Library	---	---	Seq.
20	Reader	ISP K* Source Library	---	---	Seq.
21	Reader	Seq. K* Source Library	---	---	Seq.
22	Reader	IDS Accessor - Part 1	/RDR/ACIDS1.S	2	Seq.
23	Reader	IDS Accessor - Part 2	/RDR/ACIDS2.S	5	Seq.
24	Reader	IDS Reader R* File	/RDR/IDS.RS	70	Seq.
25	Reader	ISP Reader R* File	/RDR/ISP.RS	60	Seq.
26	Reader	Seq. Reader R* File	/RDR/SEQ.RS	60	Seq.
27	Reader	DRT DBTF	/RDR/DRT.AT	1	Seq.
28	Reader	Reader Internal File	/RDR/INTRN.D	1	Seq.
29	Restructurer	Prototype JCL	/REST/PROTO.J	3	Seq.
30	Restructurer	K* Source Library	---	---	Seq.

Figure G-1 (cont'd)

System Generation Tape			User Permanent File		
File Number	Module	Contents	Suggested Name	Size (11inks)	Mode
31	Restructurer	R* File	/REST/REST.RS	86	Seq.
32	Writer	Prototype JCL	/WTR/PROTO.J	3	Seq.
33	Writer	K* Source Library	---	---	Seq.
34	Writer	COBOL Source - Part 1	/WTR/COB1.S	3	Seq.
35	Writer	COBOL Source - Part 2	/WTR/COB2.S	4	Seq.
36	Writer	R* File	/WTR/WRITE.RS	25	Seq.
37	---	Translation Library K* Source	---	---	Seq.
38	---	Translation Library Library	/UTIL/TRANS.LR	336	Rand.
39	---	DDLWT Work Database	/UTIL/WRKDB.AF	144	Rand.

Figure G-1 (cont'd)

Suffix	File content
.IS	ISP database
.J	JCL
.KS	K* source library
.LR	Random library
.MD	IDS MD section
.PM	IDSAN run-time parameter file
.RS	R* file
.S	Source code
.SQ	Sequential database
.SR	Source RIF
.ST	SDDL tables
.TD	TDL description
.TR	Target RIF
.TT	TDL tables
.61	Extended IDS MD section

4. Copy all files from the tape to permanent files. Note that no K* files are needed to run the Data Translator. Sample JCL to copy the files is in the first file on the tape. The two random files should be brought off the tape using the RREST Utility command. The Reader and Writer's source code (files 22, 23, 34, 35) should be converted from BCD to ASCII using BCDASC.

5. Convert all JCL files to ASCII and edit the JCL for each module so the proper files are referenced. See Sections 5, 6, 7, 8, 9 for the changes to be made in each JCL file. Each module can be run after completing its checklist in the User Manual.